

Formalising cartographic generalisation knowledge in an ontology to support on-demand mapping

Thesis submitted to the Manchester Metropolitan University by

Nicholas Gould

in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

Supervisory team:

Dr. Jianquan Cheng

Prof. Steve Hoon

Dr. Darren Dancey

June 2014

“In that Empire, the Art of Cartography attained such Perfection that the map of a single Province occupied the entirety of a City, and the map of the Empire, the entirety of a Province. In time, those Unconscionable Maps no longer satisfied, and the Cartographers Guilds struck a Map of the Empire whose size was that of the Empire, and which coincided point for point with it. The following Generations, who were not so fond of the Study of Cartography as their Forebears had been, saw that that vast Map was Useless, and not without some Pitiableness was it, that they delivered it up to the Inclemencies of Sun and Winters. In the Deserts of the West, still today, there are Tattered Ruins of that Map, inhabited by Animals and Beggars; in all the Land there is no other Relic of the Disciplines of Geography.”

On Exactitude in Science, Jorge Luis Borges (1998)

“A message to mapmakers: highways are not painted red, rivers don’t have county lines running down the middle, and you can’t see contour lines on a mountain.”

William Kent, Data and Reality (2000)

Abstract

This thesis proposes that on-demand mapping - where the user can choose the geographic features to map and the scale at which to map them - can be supported by formalising, and making explicit, cartographic generalisation knowledge in an ontology. The aim was to capture the semantics of generalisation, in the form of declarative knowledge, in an ontology so that it could be used by an on-demand mapping system to make decisions about what generalisation algorithms are required to resolve a given map condition, such as feature congestion, caused by a change in scale.

The lack of a suitable methodology for designing an *application* ontology was identified and remedied by the development of a new methodology that was a hybrid of existing *domain* ontology design methodologies. Using this methodology an ontology that described not only the geographic features but also the concepts of generalisation such as geometric conditions, operators and algorithms was built. A key part of the evaluation phase of the methodology was the implementation of the ontology in a prototype on-demand mapping system.

The prototype system was used successfully to map road accidents and the underlying road network at three different scales. A major barrier to on-demand mapping is the need to automatically provide parameter values for generalisation algorithms. A set of measure algorithms were developed to identify the geometric conditions in the features, caused by a change in scale. From this a Degree of Generalisation (DoG) is calculated, which represents the “amount” of generalisation required. The DoG is used as an input to a number of bespoke generalisation algorithms. In particular a road network pruning algorithm was developed that respected the relationship between accidents and road segments. The development of bespoke algorithms is not a sustainable solution and a method for employing the DoG concept with existing generalisation algorithms is required.

Consideration was given to how the ontology-driven prototype on-demand mapping system could be extended to use cases other than mapping road accidents and a need for collaboration with domain experts on an ontology for generalisation was identified. Although further testing using different uses cases is required, this work has demonstrated that an ontological approach to on-demand mapping has promise.

Acknowledgements

I would like, firstly, to thank my original Director of Studies, Omair Chaudhry for providing the inspiration for the project and giving me the opportunity to embark on a PhD. His successor, Jianquan Cheng, had the unenviable task of taking over the project half-way through and his drive, guidance, and attention to detail has ensured the completion of the project. Thanks are also due to other members of my supervisory team at MMU. The Computer Science perspective provided by Martin Stanton and his successor, Darren Dancey, was much appreciated. My supervisor, Steve Hoon, provided a wise and steady hand as he lived through the departure of a Director of Studies, a supervisor and an external advisor.

Thanks are due to the Ordnance Survey, not only for their financial contribution, but far more importantly, the input from their Research Team, led by Glen Hart and including Patrick Revell, Sheng Zhou and Stuart Thom. I would particularly like to thank two former members of the research team, Sandrine Balley and Nicolas Regnauld. Nicolas acted as my external advisor and, with his vast experience in generalisation, often helped to solve seemingly difficult problems in minutes. One of my main motivations for taking on this project was the involvement of the Ordnance Survey and my frequent visits to Southampton were a highlight of the project.

I attended the annual workshop of the International Cartographic Association's Commission on Generalisation and Multiple Representation three times during the project and returned from each workshop inspired, having received much useful advice from attendees and reviewers.

I owe a great debt to William Mackaness of Edinburgh University. In frequent meetings in Edinburgh he provided inspiration and guidance and helped me see the context of my work within the long history of generalisation. I invariably spent the train journeys back to Manchester desperately writing up my meeting notes to ensure I didn't lose any one of his wise words.

I would like to thank my wife Ann and my daughter Hannah for their forbearance and support through what has been a long three and a half years. Finally, I would like to dedicate this thesis to my father who, when told I was considering giving up a well paid job to become a student again, responded with nothing but encouragement.

Contents

List of Figures	viii
List of Tables	xiii
List of Equations	xiii
Abbreviations	xiv
1 Introduction.....	1
1.1 The importance of scale	1
1.2 The need for on-demand mapping	2
1.3 Problem statement.....	6
1.4 A knowledge engineering perspective	6
1.4.1 Knowledge engineering	6
1.4.2 A model for generalisation	7
1.4.3 The mapping engine.....	8
1.4.4 Measures	8
1.5 Aim and objectives.....	9
1.6 Scope	10
1.7 Structure of thesis.....	11
2 Knowledge representation in automatic generalisation	12
2.1 Introduction	12
2.2 Why is automatic generalisation difficult?	12
2.3 The age of algorithms.....	13
2.4 Knowledge representation in rule-based systems	13
2.5 Knowledge representation in constraints-based generalisation	14
2.6 Approaches to on-demand mapping.....	18
2.7 Taxonomies of generalisation operators	22
2.8 Ontologies for generalisation	25
2.9 Conclusions	26
3 An ontological approach to on-demand mapping.....	27

3.1	Introduction.....	27
3.2	A model for generalisation.....	27
3.3	Declarative versus procedural knowledge	29
3.4	What is an ontology?.....	30
3.5	Why use an ontology?.....	32
3.6	What type of ontology is required?.....	34
3.7	Selecting an ontology design methodology	34
3.7.1	Existing design methodologies	34
3.7.2	Formalising the ontology in Web Ontology Language (OWL).....	39
3.7.3	Evaluation	40
3.7.4	The final methodology.....	41
3.8	Conclusions.....	41
4	Reasoning with the ontology	42
4.1	Introduction.....	42
4.2	Implementing the generalisation model of McMaster and Shea.....	42
4.3	A methodology for designing the reasoning agent	44
4.3.1	Task layer.....	46
4.3.2	Inference layer – Problem-solving methods	47
4.3.3	Domain layer	50
4.4	Parameter selection using the Degree of Generalisation concept	51
4.4.1	The problem with parameters	51
4.4.2	Possible solutions to the problem of automatic parameterisation	53
4.4.3	Map evaluation	53
4.4.4	Calculating a Degree of Generalisation	55
4.5	The on-demand mapping system – system design model.....	56
4.6	Conclusions.....	60
5	Building the ontology	62
5.1	Introduction.....	62

5.2	Defining the scope of the ontology	63
5.2.1	Defining the use case	63
5.2.2	Competency questions	68
5.3	Informal conceptualisation.....	69
5.3.1	New knowledge from old	71
5.3.2	Refining the competency questions	74
5.4	Semi-formal conceptualisation.....	74
5.4.1	Modelling feature types in the ontology	77
5.4.2	Modelling operators in the ontology	79
5.4.3	Modelling transformation algorithms	84
5.4.4	Modelling measure algorithms	89
5.5	Use-case specific challenges in semi-formal conceptualisation	90
5.5.1	Selection by Attribute	90
5.5.2	Amalgamation.....	92
5.5.3	Aggregation	95
5.5.4	Pruning.....	96
5.5.5	Selecting the correct measure algorithm.....	97
5.5.6	Modelling spatial relations in the ontology	98
5.6	Formal conceptualisation	103
5.7	Evaluation	103
5.8	Conclusions	105
6	Implementing the on-demand mapping system	109
6.1	Introduction	109
6.2	Data	109
6.2.1	Accidents	110
6.2.2	Roads	110
6.3	The measure algorithms – identifying problem features.....	111
6.3.1	Measuring congestion in accident features.....	111

6.3.2	Measuring congestion in road segment features.....	115
6.4	Calculating the Degree of Generalisation	118
6.5	Algorithms for generalising the accidents	119
6.5.1	Selection by attribute	119
6.5.2	Amalgamation of accidents	122
6.5.3	Aggregation of accidents	125
6.6	Algorithms for generalising the road network	128
6.6.1	Collapse	128
6.6.2	Pruning the road network.....	129
6.7	Platform design	137
6.7.1	Interacting with the ontology - the OWL API	138
6.7.2	Implementing the Problem Solving Methods	138
6.7.3	Workflow – control knowledge	138
6.8	Results.....	145
6.8.1	Large scale – Highways engineer	145
6.8.2	Medium scale – Parent.....	146
6.8.3	Small scale – Road safety expert	146
6.9	Conclusions.....	161
6.9.1	Methodology assessment	161
6.9.2	Evaluation of the measures and the Degree of Generalisation concept.....	162
6.9.3	Evaluation of the transformation algorithms	163
7	An evaluation of the ontological approach	165
7.1	Introduction.....	165
7.2	Outstanding issues.....	165
7.2.1	Knowledge representation	165
7.2.2	Refining the ontology design.....	166
7.2.3	User requirements	168
7.2.4	Scale.....	170

7.3	Beyond the use case	174
7.3.1	Extending the current use case.....	174
7.3.2	Mapping bus routes.....	178
7.3.3	Assessing a condition other than congestion	183
7.4	Summary	188
8	Conclusions and further work.....	190
8.1	Thesis summary	190
8.2	Further work: beyond the McMaster and Shea Model.....	193
8.2.1	Supporting existing generalisation systems	193
8.2.2	Supporting web services	194
8.3	Major achievements	196
8.4	Final words.....	197
	References.....	199
	Appendix A.....	213
	Appendix B	220
	Appendix C	229
	Appendix D.....	239
	Appendix E	250

List of Figures

Figure 1.1 Road features at 1:60K (Data © Crown Copyright 2014. An Ordnance Survey/EDINA supplied service.)	1
Figure 1.2 Urban areas at different scales. Maps © Crown Copyright 2014. An Ordnance Survey/EDINA supplied service.....	3
Figure 1.3 Bicycle routes overlaid on a Google Maps background	4
Figure 1.4 Generalised road network and accidents	5
Figure 1.5 Road network and accidents hot-spots at small scale	5
Figure 1.6 High level components of an on-demand mapping system.....	8
Figure 2.1 Simple constraints	15
Figure 2.2 Complex constraint	15
Figure 2.3 Spectrum of knowledge.....	16
Figure 2.4 Automatic generalisation (Data © Crown Copyright 2014. An Ordnance Survey/EDINA supplied service.)	17
Figure 2.5 A cluster of buildings at three different scales (based on Hampe et al. (2003))	19
Figure 2.6 Comparison of three generalisation operator taxonomies.....	22
Figure 2.7 A partial taxonomy of generalisation algorithms based on Li (2006).....	24
Figure 3.1 The McMaster and Shea generalisation model modified for on-demand mapping	29
Figure 3.2 Classifying knowledge representations	31
Figure 3.3 Example relationships in an ontology	32
Figure 3.4 Comparable steps in four ontology design methodologies	36
Figure 3.5 Three phase conceptualisation in ontology design.....	37
Figure 3.6 Hybrid ontology design methodology.....	41
Figure 4.1 The process for generalising a feature collection.....	43
Figure 4.2 The role of the mapping engine.....	44
Figure 4.3 CommonKADS Expertise model (based on Schreiber et al., 1994 and Studer et al., 1998)	45
Figure 4.4 Hierarchy of tasks for mapping road accidents at a particular scale	46
Figure 4.5 Task specification for identifying a measure algorithm.....	47
Figure 4.6 Components of a PSM (based on Gómez Pérez and Benjamins, 1999)	47
Figure 4.7 Heuristic Classification PSM used for medical diagnosis (based on Studer et al, 1998)	48
Figure 4.8 Heuristic Classification PSM used for identifying the presence of a geometric condition	49

Figure 4.9 PSM for identifying a measure algorithm	49
Figure 4.10 PSM for identifying a transformation algorithm (adapted from Mustiere, 2005).....	50
Figure 4.11 Parameters for four point-aggregation algorithms	52
Figure 4.12 Legibility as a function of symbolisation	54
Figure 4.13 Feature collection definitions	55
Figure 4.14 On-demand mapping system (Balley and Regnauld, 2011a)	57
Figure 4.15 CommonKADS system design model (based on Kingston, 1998 and Schreiber et al., 1994)	58
Figure 4.16 Integrated task layer for mapping a single feature set	59
Figure 4.17 On-demand mapping system architecture design.....	60
Figure 5.1 Methodology steps with related chapter sections.....	62
Figure 5.2 Road accidents at 1:60K (data provided by Transport for Greater Manchester)	65
Figure 5.3 Resolving congestion in accident data by reducing feature density.....	66
Figure 5.4 Reducing feature congestion in road segments by reducing feature density	67
Figure 5.5 Loss of context for the accidents	68
Figure 5.6 Reducing illegibility	73
Figure 5.7 Colour coding for high level concepts	75
Figure 5.8 Selecting an algorithm – general case	76
Figure 5.9 Selecting an algorithm – particular case.....	76
Figure 5.10 The feature type class hierarchy	77
Figure 5.11 Extended feature type class hierarchy	78
Figure 5.12 Property relations between concepts	79
Figure 5.13 Selected characteristics from operator taxonomies	82
Figure 5.14 Semi-formal conceptualisation of the <i>collapse</i> operator	83
Figure 5.15 The definition of a generic transformation algorithm	88
Figure 5.16 The definition of two line simplification algorithms.....	89
Figure 5.17 The definition of a generic measure algorithm	89
Figure 5.18 Defining the selection by attribute operator	91
Figure 5.19 Preventing the application of <i>selection by attribute</i> to road features	92
Figure 5.20 The definition of the amalgamation operator	93
Figure 5.21 High density topographic features.....	93
Figure 5.22 Preventing the application of <i>amalgamation</i> to road features.....	94
Figure 5.23 A property chain.....	94
Figure 5.24 Indirect and direct restrictions on operators	95

Figure 5.25 Definition of the aggregation operator	95
Figure 5.26 Location of properties	96
Figure 5.27 The definition of the <i>pruning</i> operator	97
Figure 5.28 Definition of measure algorithms for the use case	98
Figure 5.29 Representing spatial relations in the ontology	100
Figure 5.30 Modelling the <i>general</i> semantic relationship between accidents and roads	100
Figure 5.31 Modelling a <i>particular</i> semantic relationship between roads and accidents	100
Figure 5.32 Spatial relation predicates between roads and accidents.....	101
Figure 5.33 Modelling the <i>contained by</i> spatial relation as it related to accidents and roads	102
Figure 5.34 Modelling the <i>intersects</i> spatial relation in the ontology	103
Figure 5.35 Definition of the AreaFeatureCollapseAlgorithm.....	104
Figure 6.1 Extent of use case data (background map © Microsoft Corporation 2014).....	110
Figure 6.2 Defining feature collection individuals	111
Figure 6.3 Accidents represented as points at different scales and symbol sizes.....	113
Figure 6.4 Problem accident features (highlighted in red)	114
Figure 6.5 Identifying high density of road features as <i>areas</i> using crossroad density.....	115
Figure 6.6 Determining a high density cluster of road segments	116
Figure 6.7 High density area road segments (highlighted).....	117
Figure 6.8 Identifying high density of road features represented as <i>lines</i> using crossroad density.....	118
Figure 6.9 Using the DoG to determine the class of features to retain.....	120
Figure 6.10 Target and actual features retained from the full accident dataset with selection by attribute.....	121
Figure 6.11 Number of accidents in each year of the sample dataset.....	121
Figure 6.12 Amalgamated accident cluster with DoG of 6 (accidents in red are those identified as problem features).	123
Figure 6.13 Accidents at scale 0.17 pixels/m amalgamated (yellow) with DoG of 9 (problem features in red)	124
Figure 6.14 Accidents at scale 0.17 pixels/m amalgamated with DoG of 5 (problem features in red).....	124
Figure 6.15 Accidents at scale 0.17 pixels/m. All problem feature collections amalgamated	125
Figure 6.16 Aggregated road accidents (problem features in red)	126
Figure 6.17 Aggregation with a DoG of 1 at scale 0.17 pixels/m	127

Figure 6.18 Amalgamation with a DoG of 1 at scale 0.17 pixels/m.....	127
Figure 6.19 Kernel density analysis of the complete set of road accidents (using the ArcMap 10.1 function).....	128
Figure 6.20 Simulating the collapse of area road features using the ITN network (lines)	129
Figure 6.21 Creating strokes using the good continuation principle	130
Figure 6.22 Strokes identified by the pruning algorithm.....	130
Figure 6.23 Three representations of a crossroad	131
Figure 6.24 Removing traffic islands to improve strokes.....	132
Figure 6.25 Algorithm for pruning the road network	135
Figure 6.26 Pruning the road network	137
Figure 6.27 On-demand mapping system platform design.....	137
Figure 6.28 Preliminary steps for the mapping engine	139
Figure 6.29 Initial view of the on-demand mapping system at scale 5.0 pixels/m (the dots represent accidents).....	141
Figure 6.30 Simplified steps in the <i>map bounds changed</i> event listener (FC = Feature Collection)	142
Figure 6.31 Flowchart for returning list of appropriate operators for the given feature collection (FC) and symptom	144
Figure 6.32 User selection of an operator/algorithm	145
Figure 6.33 Scale of 8.0 pixels/m no problem accident or road features identified	149
Figure 6.34 Scale 1.0 pixels/m. Problem accidents highlighted in red. (road line features dataset chosen at start)	150
Figure 6.35 Scale 1.0 pixels/m. Accidents aggregated with a DoG of 3 (suggested values of 6). No problem road features identified.....	151
Figure 6.36 Scale 0.94 pixels/m Problem accident features highlighted in red.....	152
Figure 6.37 Scale 0.94 pixels/m. Selection By Attribute applied to accidents with a DoG of 9 (suggested value was 4)	153
Figure 6.38 Scale 0.15 pixels/m. Problem accident features highlighted in red.....	154
Figure 6.39 Scale 0.15 pixels/m. Problem road area features (MasterMap) highlighted in red	155
Figure 6.40 Scale 0.15 pixels/m. Problem road line features (ITN) after collapse highlighted in red	156
Figure 6.41 Scale 0.15 pixels/m. Accidents amalgamated with a DoG of 9 (suggested value was 9). Roads pruned with DoG of 4 (suggested value was 4)	157

Figure 6.42 Scale 0.15 pixels/m. Accidents amalgamated with a DoG of 5 (suggested value was 9). Roads pruned with DoG of 4 (suggested value was 4)	158
Figure 6.43 Scale 0.15 pixels/m. Accidents aggregated with a DoG of 5 (suggested value was 9). Roads pruned with DoG of 7 (suggested value was 4)	159
Figure 6.44 Scale 1.69 pixels/m. Accidents aggregated with a DoG of 5 (suggested value of 5)	160
Figure 6.45 Compromise between showing all of the accident clusters and reducing road density	163
Figure 7.1 Example icons for generalisation operators	169
Figure 7.2 The relation between feature type and role	170
Figure 7.3 Road accidents in Greater Manchester 1994 to 2008.....	171
Figure 7.4 Applying displacement.....	172
Figure 7.5 London Underground map (fragment) © Transport for London	173
Figure 7.6 Accidents at a junction with traffic lights and bus stops.....	175
Figure 7.7 Levels of abstraction for different feature types	176
Figure 7.8 Conceptual and structural operators	177
Figure 7.9 Amalgamating two features	178
Figure 7.10 Bus routes in Greater Manchester	179
Figure 7.11 A bus route (in red) at the neighbourhood scale mapped with the road network.....	180
Figure 7.12 Two intersecting bus routes with bus stops.....	183
Figure 7.13 A building represented at two different scales	184
Figure 7.14 The effect of building shape on perceptibility	185
Figure 7.15 The application of the Douglas-Peucker (DP) algorithm with different tolerances	187
Figure 8.1 Semantic injection of generalisation web services (based on Janowicz et al., 2010)	195
Figure 8.2 The webProtégé interface.....	196

List of Tables

Table 1.1 Example relative scales for mapping road accidents	6
Table 2.1 Approaches to on-demand mapping	21
Table 5.1 Potential users of an on-demand mapping system.....	64
Table 5.2 Example concept descriptions	71
Table 5.3 Operator characteristics	81
Table 5.4 Characteristics of generalisation algorithms.....	85
Table 5.5 Characteristics of algorithm parameters	86
Table 5.6 Example algorithm characterisation	87
Table 6.1 Parameters for the <code>findHighPointDensityClusters</code> measure algorithm.....	113
Table 6.2 Parameters for the <i>findHighDensityAreaCrossroadClusters</i>	116
Table 6.3 Parameters for the select by attribute algorithm	120
Table 6.4 Parameters for the point amalgamation algorithm.....	122
Table 6.5 Parameters for the basic pruning algorithm.....	133
Table 6.6 Parameters for the pruning with respect to accidents algorithm.....	136
Table 6.7 Parameters for function to identify candidate operators for a particular symptom	143
Table 7.1 Higher level collective concepts for different feature types	176

List of Equations

Equation 4.1	55
Equation 6.1	112
Equation 6.2	115
Equation 6.3	119
Equation 6.4	133

Abbreviations

API	Application Programming Interface
CRS	Coordinate Reference System
DoG	Degree of Generalisation
FC	Feature Collection
GIS	Geographic Information System
GML	Geographic Markup Language
ITN	Integrated Transport Network
KBS	Knowledge-Based System
NMA	National Mapping Agency/Authority
OGC	Open Geospatial Consortium
OS	Ordnance Survey
OWL	Web Ontology Language
PSM	Problem Solving Method

1 Introduction

1.1 The importance of scale

Many governments, both national and local, are increasingly making spatial data freely available (Janssen et al., 2012; Tinati et al., 2012). For example, the DataGM website provides access to a variety of georeferenced datasets including road traffic accidents, fire and rescue incidents, bus stops, bus routes and traffic signals in Greater Manchester (Trafford Council, 2012). There is no single ideal scale at which to study any geographical phenomenon (Mackaness, 2007) and mapping this data at multiple scales, with topographic features providing context, will aid interpretation of the data. At a large scale we can glean the attributes of individuals; at a small scale we can put the individuals in context (Mackaness, 2007).

However, simply changing the scale at which data is viewed is not sufficient. At smaller scales the map may suffer from feature *congestion*, for example, which causes illegibility (Figure 1.1).

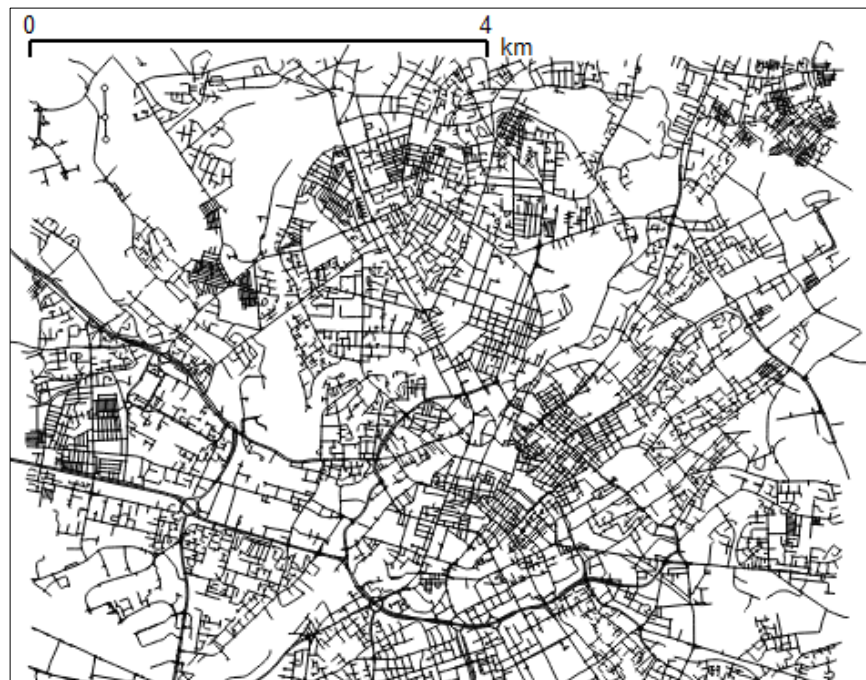


Figure 1.1 Road features at 1:60K (Data © Crown Copyright 2014. An Ordnance Survey/EDINA supplied service.)

To ensure a legible map, the representation of data at different scales requires *generalisation*, defined by the International Cartographic Association as “the selection and simplified representation of detail appropriate to scale and/or purpose of a map” (International Cartographic Association, 1973). Generalisation is performed by a number of generalisation

operators which can be illustrated by considering the depiction of a superstore at a number of different scales (Figure 1.2). As the scale decreases detail is lost but context is gained: “we can see different patterns, different relationships, between different entities” (Mackaness et al., 2014, p5).

Scale is hard to define and is linked to a number of related concepts such as resolution and accuracy but what is traditionally meant by scale is the *cartographic ratio*: the ratio of a distance on the map to the distance on the ground (Li, 2006). The introduction of devices to display maps, such as laptops and smart phones, has made the definition more difficult still and a definition appropriate to this research will be offered later (section 6.3.1).

1.2 The need for on-demand mapping

The maps displayed in Figure 1.2 were designed by a National Mapping Agency (NMA), in this case the Ordnance Survey, to serve a number of purposes. However, the digitisation of map features and the automation of map production provides an opportunity to construct maps that suit a particular user’s needs and are therefore more effective (Wilson et al., 2010). For instance, route finding using a map designed for route finding is likely to be more successful than using a generic multi-purpose map (Matsuo et al., 2011). This will require on-demand mapping: “the creation of a cartographic product upon a user request appropriate to its scale and purpose” (Cecconi, 2003, p17). This definition should be modified to specify that the user need not be a cartographer.

As stated, generalisation is performed by a number of operators. However, an operator may be implemented by a number of algorithms. For example, the outline of the superstore in Figure 1.2a was simplified, by the removal of selected points, to produce the outline in Figure 1.2b. This could have been performed by a number of algorithms (Kulik et al., 2005; Wang & Müller, 1998; Visvalingam & Whyatt, 1993; Muller, 1987; Douglas & Peucker, 1973).

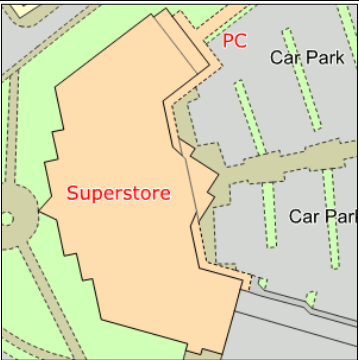
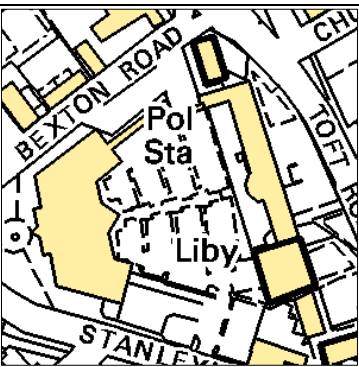
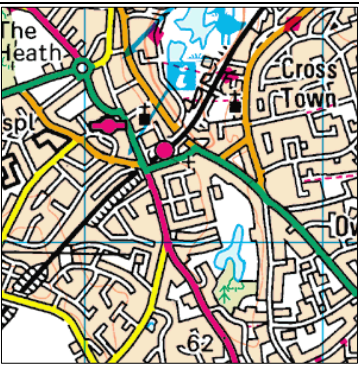
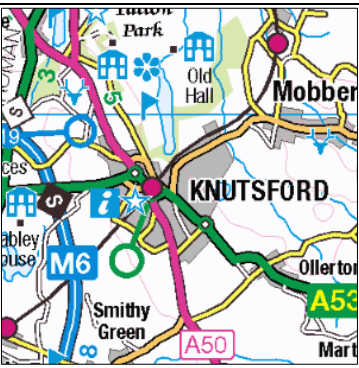

	<p>(a) 1 to 5000</p> <p>The outline of the superstore and its car park are clearly visible and at this scale a relatively accurate measure of the building footprint is possible.</p>
	<p>(b) 1 to 10000</p> <p>The building outline has been <i>simplified</i> and the level of detail reduced. However, some of the neighbouring buildings and streets are now visible and it can be seen that it is not an “out-of-town” development.</p>
	<p>(c) 1 to 50000</p> <p>The superstore is no longer visible. Individual buildings are no longer depicted (apart from churches which have been <i>symbolised</i>) and have been <i>amalgamated</i> into built-up areas.</p>
	<p>(d) 1 to 250000</p> <p>All of the buildings are depicted as a single shaded area but it is now possible to see that the superstore is part of an urban area called Knutsford. Although the road network has been <i>pruned</i>, and only the major roads in Knutsford are depicted, it can be seen that Knutsford is near a motorway network and it is also possible to see some of the neighbouring towns connected by rail.</p>
	<p>(e) 1 to 1000000</p> <p>All detail of Knutsford is lost and it has been <i>collapsed</i> to a point. However, we can now see that Knutsford is to the south of a major conurbation (Manchester). Further reductions in scale would eventually reveal that the superstore is on an island.</p>

Figure 1.2 Urban areas at different scales. Maps © Crown Copyright 2014. An Ordnance Survey/EDINA supplied service

Much progress has been made in recent years in the automated production of cartographic products. For example, the Netherlands' Kadaster recently developed a process that created a 1:50K topographic map without any human interaction (Stoter et al., 2014). However, the complex workflow required to generate the map was produced manually. If we assume that an on-demand mapping system will be utilised by the non-cartographer then the process of selecting, sequencing and execution of generalisation algorithms should be a completely automatic process.

Existing automatic generalisation systems are designed to produce general purpose maps at a limited range of target scales (Figure 1.2) and portraying familiar topographic features such as buildings, roads and rivers (Stanislawski & Savino, 2011; Yan et al., 2008; Chaudhry & Mackaness, 2005). An on-demand mapping system should be able to map any feature type at any scale. For a given operator, different algorithms may specialise in different feature types. For example, the algorithms used to prune *river* networks (Stanislawski & Savino, 2011; Touya, 2007) differ from those used to prune *road* networks (Benz & Weibel, 2013; Liu et al., 2010; Touya, 2010). This implies that the mapping of some of the feature types mentioned earlier, such as road accidents, will also require special consideration.

Another requirement of an on-demand mapping system is that it should be able to integrate user-supplied data with base data that provide context. A simple overlay of user data on top of a generic base map is not sufficient. This is often a problem with web-based “mashups” that use tools such as the Google Maps API to produce interactive maps (Batty et al., 2010). For example, the road name labels in Figure 1.3 are obscured by the bicycle routes.

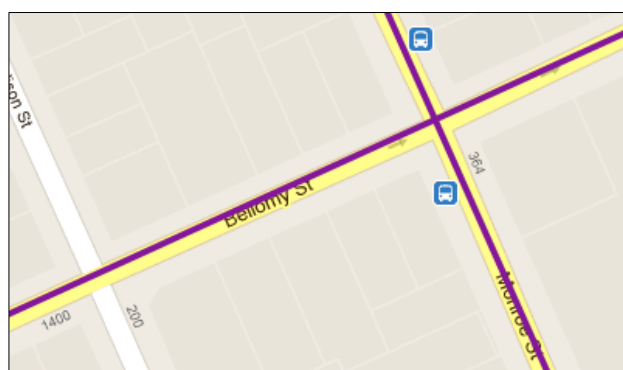


Figure 1.3 Bicycle routes overlaid on a Google Maps background

The need for integration is further highlighted when we consider the problem of mapping road accidents where generalisation of the road network may lead to loss of context for the accident data. For example, the roundabout in Figure 1.4a has been collapsed to a point (Figure 1.4b) and the accidents on the roundabout have lost their context.

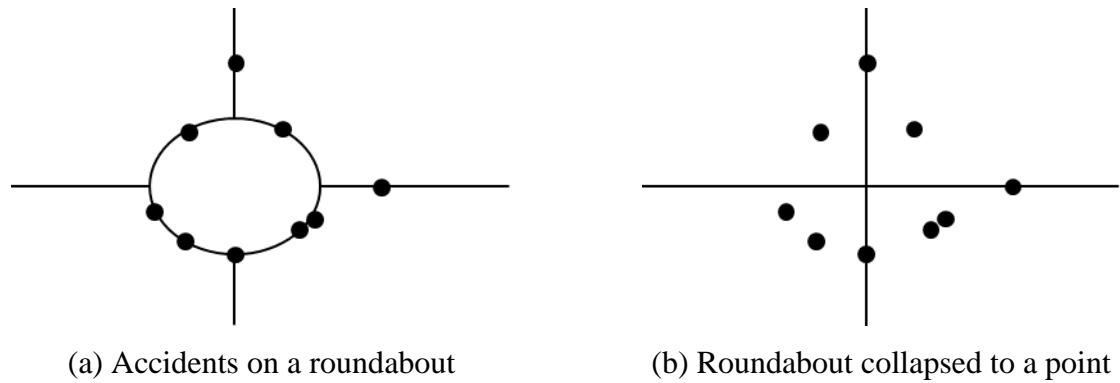


Figure 1.4 Generalised road network and accidents

The possible loss of context is less obvious when the roads and accidents are considered at a smaller scale (Figure 1.5). In this example the individual accidents have been grouped to form polygons. The removal of the minor road (A), caused by the change in scale, will lead to the loss of context and the accident hot-spot at the junction will appear to be positioned part way along a road segment.

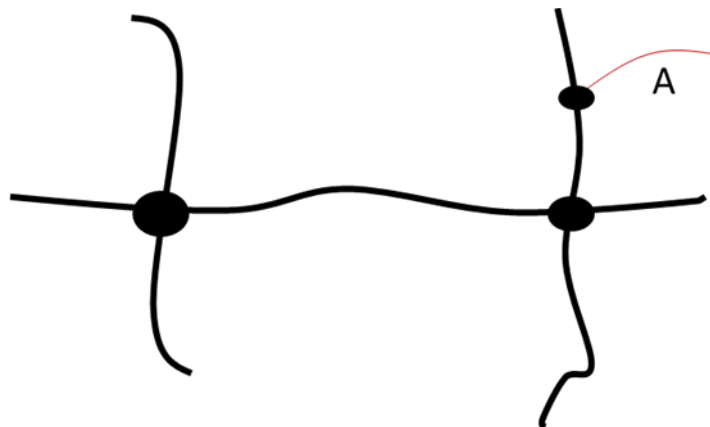


Figure 1.5 Road network and accidents hot-spots at small scale

The usefulness of generalisation is not limited to cartographic legibility. The generalisation of an urban street network can be used to understand the structure, function and organisation of a city (Jiang & Claramunt, 2004), for example by allowing the modelling of traffic flows (Jiang & Liu, 2009). Maps such as that in Figure 1.5 could, for example, be used to allow a parent to plot a safe walking route to school for a child. However, the scale has to be appropriate (Table 1.1).

Scale	Extent	User	Purpose
Large	Road junction	Highways engineer	To identify the problem arm(s) of a junction
Medium	Local neighbourhood	Parent	To identify safe routes to school
Small	Local authority	Road safety expert	To identify accident hot-spots

Table 1.1 Example relative scales for mapping road accidents

1.3 Problem statement

On-demand mapping requires fully automatic generalisation. Generalisation algorithms have been developed and refined since the 1960s (Li, 2007) and since the 1980s ways have been sought to automate their selection, sequencing and execution rather than simply execute a pre-defined sequence or “batch” (Harrie & Weibel, 2007).

The difficulty of the problem can be illustrated by examining a small part of the workflow designed to produce the map of the Netherlands mentioned earlier (Stoter et al., 2014). The decision was made to thin, or prune, the waterways network using a *road* network thinning algorithm rather than one of the algorithms for thinning natural *hydrographic* networks because the Dutch water network is almost entirely man-made and resembles a road network. How could such a seemingly illogical decision be made by an on-demand mapping system?

Following selection and sequencing, execution of the algorithms is required. Most generalisation algorithms will have one or more parameters and parameter value selection has been identified as having a major impact on the results of generalisation (McMaster & Shea, 1992). However, even algorithms that implement the same operator rarely have matching parameters and it is unrealistic to expect non-expert users to provide parameter values.

1.4 A knowledge engineering perspective

1.4.1 Knowledge engineering

The *knowledge* required to automatically produce the general purpose maps described earlier is either embedded in software or possessed by the NMA experts that configure the software. If we consider agent-based generalisation systems, their associated knowledge base has to be updated every time a new generalisation algorithm is introduced to the system or when user requirements change (Taillandier & Taillandier, 2012). Carral et al. (2013) make a more general point that the knowledge relating to scale dependency in digital representations is informally specified and hidden in application source code. This is not conducive to on-demand mapping. If the user wishes to map an unfamiliar feature type then the knowledge

required to generalise those features has to be encapsulated in the system. It would be preferable to capture that knowledge in a representation that could be shared.

The intention is to consider on-demand mapping as a knowledge engineering problem. Knowledge engineering has historically been seen as a human to machine *transfer process* and more recently as a *modelling process* (Studer et al., 1998). The focus of this research is on the latter. Acquisition of generalisation knowledge has been the focus of a number of studies (Mustiere, 2005; Kilpeläinen, 2000; Chang & McMaster, 1993) but there has been less discussion of the techniques for representing generalisation knowledge.

There have been a number of attempts to explicitly represent generalisation knowledge in taxonomies of operators (Roth et al., 2011) but these have invariably been informal, natural language representations. If we are to *reason* about the process of generalisation then formal, machine-readable representations are required.

On-demand mapping requires the integration of thematic data with topographic data and to do this automatically requires a machine-readable representation of the *semantics* of the features and the *relationships* between them (Stoter et al., 2010; Wolf, 2009; Dutton & Edwardes, 2006). One way of encapsulating the semantics of a domain is by using an *ontology* (Kavouras & Kokla, 2008). However, the aim is not simply to describe geographic features in an ontology but also the *process* of generalisation and then to use reasoning to infer the operators and algorithms required to resolve particular map conditions, such as feature congestion.

1.4.2 A model for generalisation

If the ontology is to be used to aid the process of generalisation, in particular the selection of generalisation operators and algorithms, then a contextual framework, or *generalisation model*, is required (Sarjakoski, 2007; McMaster, 1991). The most appropriate model for on-demand mapping is that of McMaster and Shea (1992) who seek to define the *Why*, *When* and *How* of generalisation.

There are a number of reasons *why* generalisation is required but the focus will be on *reducing complexity*. The moment *when* to generalise is when certain geometric conditions such as *congestion* or *imperceptibility* appear in the mapped data (Figure 1.1). *How* to generalise is determined by the operators.

The model describes three transformation controls, *generalisation operator selection*, *algorithm selection* and *parameter selection*. The aim of this research is to use the ontology to reason about the first two of these controls.

Early knowledge-based systems for generalisation were rule-based (Buttenfield & McMaster, 1991). Rules represent *procedural* or “how-to” knowledge, but the intention here is to represent generalisation as *declarative* or “know-what” knowledge which seeks to make statements or declarations about the world (Genesereth & Nilsson, 1998). Declarative knowledge has the advantage of being extendable by inference, providing “new expressions from old” (Davis et al., 1993).

1.4.3 The mapping engine

An ontology is a knowledge representation and as such is useful for “reasoning about the world rather than taking action in it” (Davis et al., 1993, p17). Furthermore, a disadvantage of the flexibility of declarative knowledge is that, in contrast to procedural knowledge, it does not define what to do with the knowledge. What is required is an on-demand *mapping engine* that will use the ontology to implement the McMaster and Shea (1992) model (Figure 1.6).

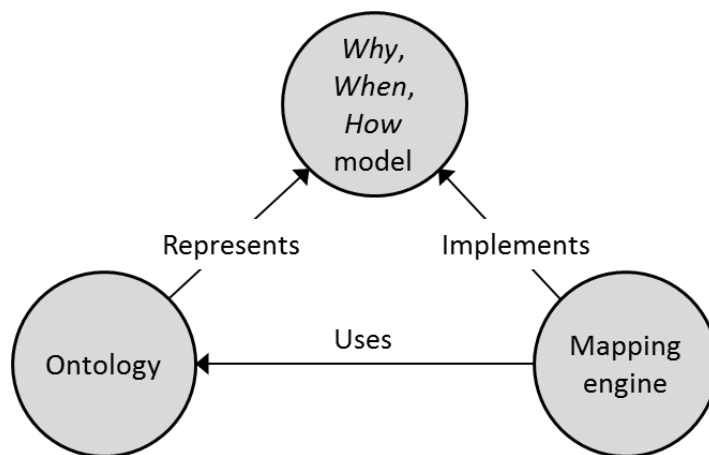


Figure 1.6 High level components of an on-demand mapping system

The mapping engine will take the user requirements, including target scale, the selected data sources and query the ontology for the appropriate operator and algorithm to employ given a particular condition. However, a mechanism is required to identify the presence of such conditions.

1.4.4 Measures

The *when* to generalise is calculated by one or more of the *spatial and holistic measures* described in the McMaster and Shea (1992) model, which will identify *decision points* (Roth

et al., 2011). For example, feature congestion can be determined by a *density measure* (Stigmar & Harrie, 2011). Measures were implemented by a number of measure algorithms.

The third transformation control in the model of McMaster and Shea (1992) is algorithm *parameter selection*. As discussed earlier, it is unrealistic to expect the non-cartographer to provide parameter values and since the system is intended to be multi-scale it will not be possible to pre-define these values, in the ontology, say. To resolve this, a *Degree of Generalisation* (Zhou & Jones, 2003) is calculated, based on the results from a set of measure algorithms, and used as a generic parameter, which determines the “amount” of generalisation required.

1.5 Aim and objectives

The aim of this research is to develop an on-demand mapping system based on an ontology. The following objectives will achieve this aim:

1. To model the process of generalisation using an ontology.
2. To devise a method for automatically selecting the appropriate algorithms for mapping geographic features at multiple scales using the ontology.
3. To devise a method for generating parameter values for the selected algorithms’ parameters based on the current conditions in the mapped data.
4. To develop an on-demand mapping prototype for road accidents and roads that will test the above methods.

The aim and objectives can be expressed as a number of **research questions**:

- How can the process of cartographic generalisation be captured in an ontology?
 - What are the essential characteristics of generalisation operators?
 - What are the essential characteristics of generalisation algorithms?
- How can knowledge of the geographic data (accidents and roads) be described in the ontology in such a way that it can be used to guide the process of on-demand mapping?
 - What are the essential characteristics of a geographic feature type that effect how features of that type are generalised?
- Can we automatically determine the conditions under which the data should be generalised?
- Can the ontology be reasoned with, using inference, to automatically select the generalisation operators and algorithms that will resolve particular conditions in the mapped data?

- Can this be done by using semantics, expressed as declarative knowledge, rather than procedural knowledge?
- Once an algorithm has been selected can values for its parameters be automatically generated?
- Can the geographic features data and ontology be combined in an on-demand mapping system?

1.6 Scope

On-demand mapping is too complex a problem to be solved by a single project (Balley & Regnauld, 2011a) and it is therefore necessary to limit the scope of this research. Based on the objectives the following topics are *within* the scope of this research:

Generalisation to ensure legibility

There are a number of reasons why generalisation might be required (McMaster & Shea, 1992) but this research will focus on maintaining legibility and, in particular, reducing feature congestion.

Mapping of roads and accidents

The research will be limited to a particular use case; the mapping of road accidents and the road network, which will provide context for the accidents. This use case was selected since it involves the mapping of both thematic and topographic data. Most automatic generalisation research has focussed solely on topographic features.

Automatically determining algorithm parameter values

One of the objectives is to automatically select appropriate generalisation algorithms. To execute the selected algorithms it is necessary to provide parameter values. The typical user cannot be expected to do this. As will be discussed in Chapter 4, to do this automatically is potentially a difficult undertaking but necessary for on-demand mapping.

The following topics are *out of the scope* of the research:

Cartographic knowledge acquisition

The focus will be on the representation of knowledge rather than how to acquire it.

Formal specification of user requirements

For an on-demand mapping system it will be necessary to formalise the user's requirements. Foerster et al. (2012), for example, propose user profiles and Hubert and Ruas (2003) look at machine learning to capture user needs. This is too large a topic to be examined fully in this

research project but Balley et al. (2014) recently provided a detailed discussion on map specifications and user requirements, which included on-demand mapping.

On-the-fly mapping

Some definitions of on-demand mapping have included “on-the-fly” mapping (Cecconi et al., 2002), where the speed at which the map is generated is important. This requirement is beyond the scope of this research.

Web services

The long term aim of the on-demand mapping project at the Ordnance Survey is to expose existing generalisation algorithms as web services. There has been considerable research into geospatial processing web services in general (Fitzner et al., 2011; Brauner et al., 2009; Friis-Christensen et al., 2007; Lemmens et al., 2007) and generalisation web services in particular (Gould, 2012; Neun et al., 2009; Foerster et al., 2008; Burghardt et al., 2005). This research will not however consider web services specifically but will focus on the means by which generalisation capability, in general, can be automatically selected.

On-demand mapping is a complex problem that requires collaboration (Balley & Regnauld, 2011a). This research is part of a collaboration between Ordnance Survey and its French equivalent, Institut National de L’Information Géographique et Forestière (IGN), (Balley et al., 2012; Touya et al., 2012; Balley & Regnauld, 2011b).

1.7 Structure of thesis

Following a background study of automatic generalisation in the context of knowledge representation (Chapter 2), the methodology section of thesis makes a case for describing generalisation knowledge in an ontology and details a methodology for building such an ontology (Chapter 3). Chapter 4 then describes a methodology for building an on-demand mapping system based on the ontology. The implementation stage of the thesis describes how the ontology was built (Chapter 5) and how the mapping engine was developed (Chapter 6). A use-case was developed to guide this stage. Chapter 7 contains a discussion of the scalability of the ontological approach to determine the ease at which it can be extended to other use cases. The thesis concludes with a summary and a discussion of further work (Chapter 8).

2 Knowledge representation in automatic generalisation

2.1 Introduction

“The content and graphical structure of a complex, demanding map image can never be rendered in a completely automatic way” (Imhof (1982) cited in McMaster and Shea (1992)).

“This is the first time that a complete topographic map has been generalized without any human interaction” (Stoter et al., 2014, p1).

It is necessary to establish the developments between 1982 and 2013 that made the impossible, possible. This chapter discusses developments in automatic generalisation, on the understanding that on-demand mapping requires automatic generalisation. The contention of this thesis is that on-demand mapping is a knowledge engineering problem and that it is necessary to represent generalisation knowledge explicitly. Knowledge engineering involves the acquisition, representation and use of knowledge for expert systems. What knowledge is required for generalisation, and how it is acquired, has been well researched (Mustiere, 2005; Kilpeläinen, 2000; Weibel et al., 1995; Chang & McMaster, 1993). However, rather than simply present an overview of previous work in automatic generalisation the focus in this chapter will be on the *types* of knowledge that generalisation systems have used and how they have represented it.

Before examining the literature it is necessary to discuss why automatic generalisation is a difficult task.

2.2 Why is automatic generalisation difficult?

The fact that there are a large number of algorithms that will implement any particular generalisation operator is evidence of the complexity of the problem of automatic generalisation.

Mackaness (2007) suggests four reasons why automatic generalisation is difficult. Firstly, the design process is complex and involves a compromise that satisfies a number of competing objectives. The importance of studying geographic phenomena at different scales was highlighted in section 1.1 but this need leads the second difficulty, how to abstract out different representations of the same source data and then assess their truth. The third difficulty is in the necessity of treating generalisation as a modelling process and not the merely the manipulation of geometric primitives. Finally, there is the need to take account of, and formalise, the user’s need. The latter is particularly important if we wish to realise on-demand mapping.

To these can be added the fact that if the source data is held in a spatial database, which itself represents a model of reality, then in creating a generalised map we are creating a model of a model.

2.3 The age of algorithms

Cartographic generalisation was for centuries the domain of cartographers, a manual process of “art clarified by science” (Eckert, 1908). The 1960s, however, saw the introduction of *digital* generalisation where algorithms were developed to automate specific generalisation tasks such as line simplification and smoothing (Li, 2007). Algorithms were combined in a process known as *batch generalisation* where sequences of algorithms and parameter values were defined in advance (Harrie & Weibel, 2007). The knowledge of cartographers was encapsulated in the choice of algorithms and their parameter values.

The inflexibility of batch processing led to a search for a method for automatic generalisation where the transformations required to produce a map were not pre-defined. A distinction is made here between *digital generalisation*, which is the use of algorithms to perform generalisation, and *automatic generalisation*, the automatic selection of those algorithms. The first experiments were in rule-based expert systems.

2.4 Knowledge representation in rule-based systems

Guidelines for cartographers were often expressed in the form of rules. The following example is from the Ordnance Survey guidelines for the portrayal of ponds at a scale of 1:10000 (cited by João, 1998, p16):

“5434: Ponds smaller than 1mm^2 will be omitted unless they are in an area where there is little other water detail, in which case they will be enlarged to the minimum size (i.e. 1mm^2)”
and

“5435: Ponds which are less than 1mm wide but at least 2mm long will be enlarged to 1mm wide if space permits.”

The above rules are not, however, machine-readable. To automate generalisation, rule-based expert systems attempted to encapsulate the above knowledge in a machine-readable form. For example, Nickerson (1991) attempted to convert the guidelines for producing the *Canadian National Topographic Series* into condition-action rules of the form:

IF <condition> THEN <action>

Exceptions such as “if space permits” can be captured in the condition-action model. However, a number of problems with the rule-based approach have been identified: the sheer number of rules required (Steiniger et al., 2010; Harrie & Weibel, 2007); the cascading effects of generalisation, which rule-based systems find difficult to control; and the problem of competition between rules (Armstrong, 1991). Beard (1991) also stressed the importance of addressing user needs in a mapping system, which makes it difficult to build a finite set of rules suitable for all requirements.

Many of these problems are a result of the fact that the condition-action rules, or *production* rules in Artificial Intelligence terms, represent *procedural knowledge*. Procedural knowledge describes “what to do, when” (Rich & Knight, 1991, p113). There is a coupling of the knowledge with a description of what to do with that knowledge. The latter is represented by the *action*. This has its advantages but procedural knowledge is hard to repurpose. A rule is required for every situation and for a dynamic system, such as on-demand mapping, the rule-based approach is unfeasible. This view was reinforced by early explorations in this research project, which involved a rule-based approach (Gould & Chaudhry, 2012) (see Appendix A).

2.5 Knowledge representation in constraints-based generalisation

Having identified a number of problems with the rule-based approach, Beard (1991) suggested a *constraints-based* approach. This approach can be summarised as: use the constraints to define the desired output and then use an *optimisation* technique to determine the best combination of *operators* to meet the constraints (Burghardt et al., 2007). For example, there might be a constraint on the building size displayed at a certain scale. At 1:50K the minimum building size might be defined as 50m², on the understanding that anything smaller will not be perceptible on a map at that scale. In contrast to the condition-action approach, this approach represents a decoupling of the map requirements and the method of achieving those requirements; a separation of conflict *analysis* and conflict *solution* (Burghardt et al., 2007). At their simplest, constraints represent *declarative* knowledge. Declarative knowledge makes statements, or declarations, about the world (Genesereth & Nilsson, 1998).

Dutton and Edwardes (2006) make a distinction between the more “simple” constraints involving minimum widths and separation distances and those that have a semantic element involving, say, the relationships between buildings and access roads and between bridges and rivers.

Consider a constraint that specifies a minimum distance between buildings; one way of satisfying this constraint would be to *displace* one or other or both of the buildings, another would be to *amalgamate* them (Figure 2.1a). Such a constraint can be considered “simple” (Dutton & Edwardes, 2006). That is, the constraint relies on a single value that acts as a threshold. We can infer that if the two buildings were too close then they would be indistinguishable but it is not made explicit why the constraint is necessary. There are no semantics associated with the constraint. Another “simple” constraint is one that specifies the minimum size of a building that can be included on the map. The building could be *eliminated* (as insignificant) or, if the building was of sufficient importance, it could be *exaggerated* (Figure 2.1b). Again we can infer why the constraint is required, if the building was too small it would be imperceptible on the map, but again the constraint lacks semantics. However, in this case, the *solution* relies on semantics. It is necessary to know the significance of the building before we decide to eliminate or exaggerate it; the solution requires further *declarative* knowledge.

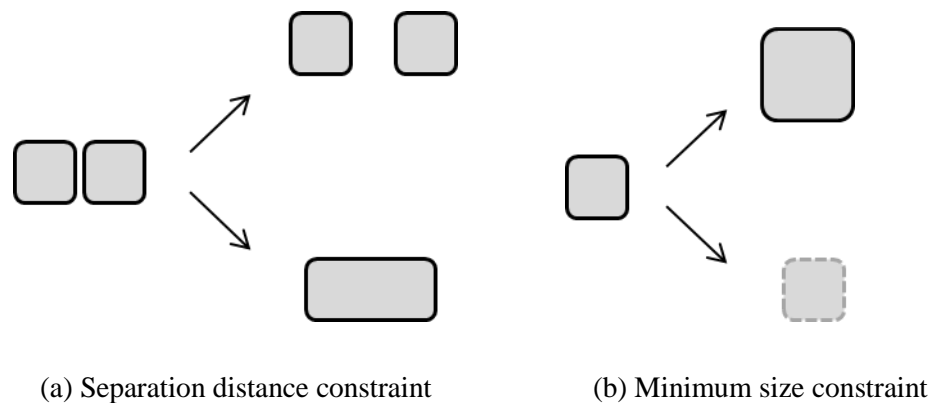


Figure 2.1 Simple constraints

Consider a more complex constraint such as “roads that provide access to buildings should be preserved” as depicted in Figure 2.2. The minor road sections *b*, *c* and *d* would normally be removed because of a minimum width constraint. However, since section *d* provides access to the building, it is preserved.

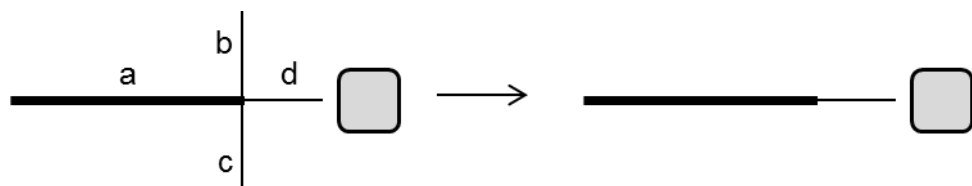


Figure 2.2 Complex constraint

We can infer from the constraint that there is a relationship between roads and buildings; roads provide *access* to buildings. At a higher level we can say that the road provides *context* for the building. This can be classified as declarative knowledge. However, it could be argued that the constraint contains some degree of “how-to” knowledge; to maintain access *preserve* the minor road. The distinction between procedural and declarative knowledge is fuzzy and can be viewed as a spectrum (Rich & Knight, 1991). Generalisation algorithms and condition-action rules clearly contain procedural knowledge (Figure 2.3) and simple constraints clearly contain declarative knowledge. The situation is less clear with more complex constraints.

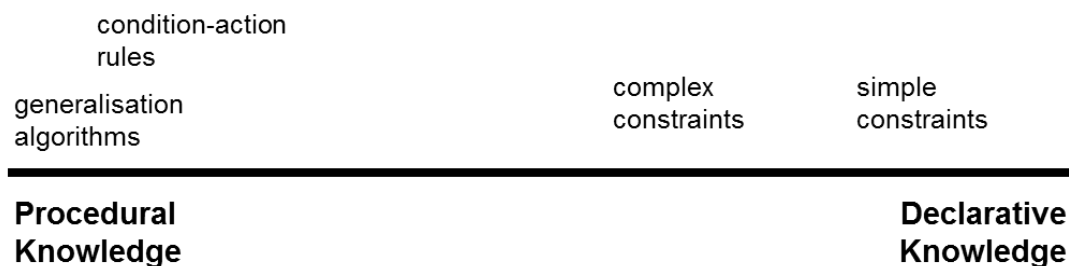


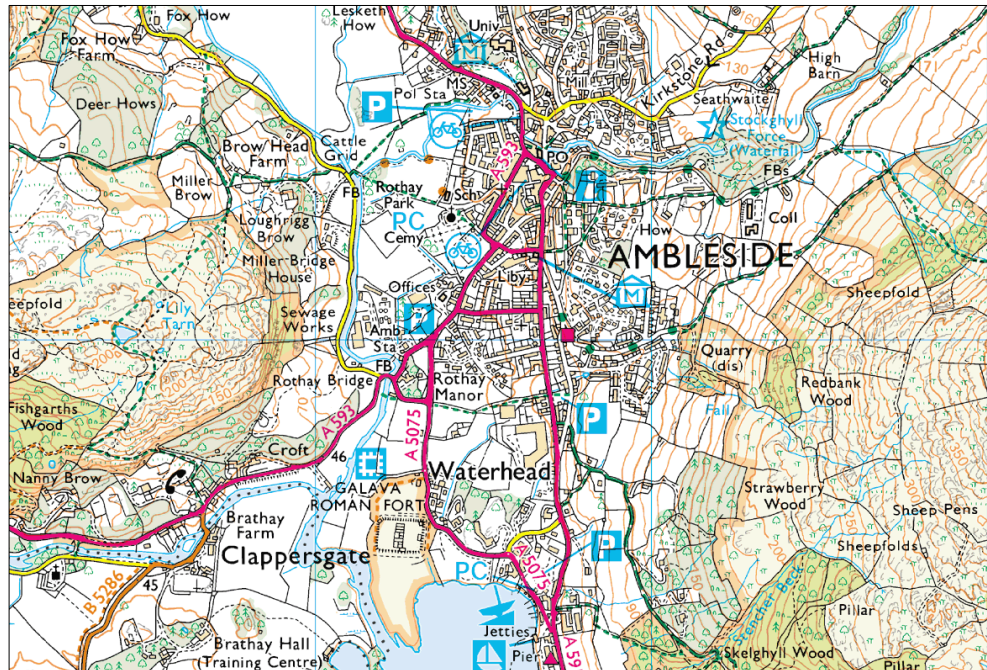
Figure 2.3 Spectrum of knowledge

There have been a number of attempts to classify constraints in a more sophisticated way than merely simple or complex. Originally Beard (1991) grouped constraints as either *graphic*, *structural*, *application* or *procedural*. More recently, Burghardt et al. (2007) defined a more formal typology in an attempt to harmonise the different constraints defined by a number of National Mapping Agencies. At the top level of their hierarchy there is an initial division between *legibility* constraints and *preservation of appearance* constraints. The typology also considers the number of features and the feature type the constraints apply to. The constraint involving roads and buildings would fall under the sub-category of *topology* in the category *preservation of appearance*.

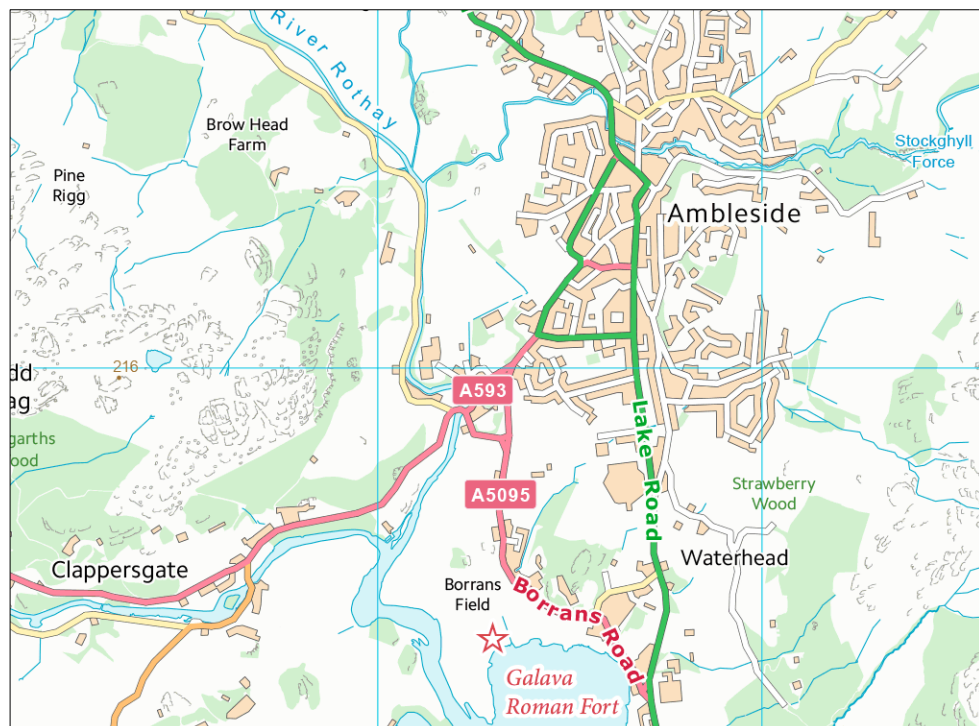
Burghardt et al. (2007) concluded that it was not possible to automatically translate the knowledge in object-oriented topographic data models into constraints since the data models lacked the ability to express such constraints as “a building must always be accessible from a road”. So the formalisation of constraints requires *human interpretation*.

Despite reservations we can say that constraints represent declarative knowledge. However, a declarative representation of knowledge requires a method of specifying what is to be done with the knowledge (Rich & Knight, 1991). This is the role of the optimisation technique, which has to ensure that the constraints are satisfied. The result is generally a compromise

since a feature type may have a number of constraints, some of which may conflict. For example, a *minimum size* constraint on two neighbouring buildings, which could lead to an increase in size, might conflict with an *overlap* constraint (on all features), which could lead to a displacement, which might conflict with a *position should not change* constraint.



(a) Standard 1:25K product



(b) Vector Map District at 1:25K

Figure 2.4 Automatic generalisation (Data © Crown Copyright 2014. An Ordnance Survey/EDINA supplied service.)

The solution to constraint conflicts is to associate a cost with each constraint violation then attempt to minimise the overall cost. A number of optimisation techniques have been developed but the agent-based method (Ruas & Duchêne, 2007; Lamy et al., 1999) is the only one that can utilise all generalisation operators (Harrie & Weibel, 2007) and is widely used in map production (Revell et al., 2011; Lecordix & Lemarie, 2007). Figure 2.4 provides an example of a map produced by an agent-based system (Figure 2.4b), in contrast to a more traditionally produced map (Figure 2.4a). In an agent-based system each topographic feature is represented by an agent. Agents can also represent groups of features (meso-agents). For each mapped feature *type* the system has to list the relevant constraints, the priority of those constraints (in case of conflicts) and the algorithms that may resolve the particular constraint (Taillandier et al., 2011). It is not always apparent what type of knowledge is being represented here. For example, the list of algorithms that resolve a particular constraint could be classed as declarative or procedural knowledge.

Agent-based systems employ a tree-based search to reach the ideal solution. This search can be optimised by using another type of knowledge, *control knowledge* (Taillandier et al., 2011). An example of control knowledge is a *stopping criterion* that determines if a search of a particular branch of the tree is worth pursuing.

The flexibility of the constraints-based technique is the reason why it has been more accepted than the rule-based approach and it should be considered for on-demand mapping. All of the knowledge described above, including the control knowledge, represents a knowledge base for the agent-system. However, this knowledge base has to be updated each time a new feature type has to be mapped or a new algorithm is incorporated into the system (Taillandier & Taillandier, 2012). Also constraints such as those that represent minimum width and minimum separation require threshold values that might not suit all map conditions. It is hard to see how the constraints-based approach can be easily extended to map a variety of feature types at multiple scales without expert involvement.

2.6 Approaches to on-demand mapping

One possible solution to on-demand mapping is the multiple representation database (MRDB) or multi-scale database (MSDB) which stores different representations of the same objects at different scales with bi-directional links between each entity at different scales (Dunkars, 2004).

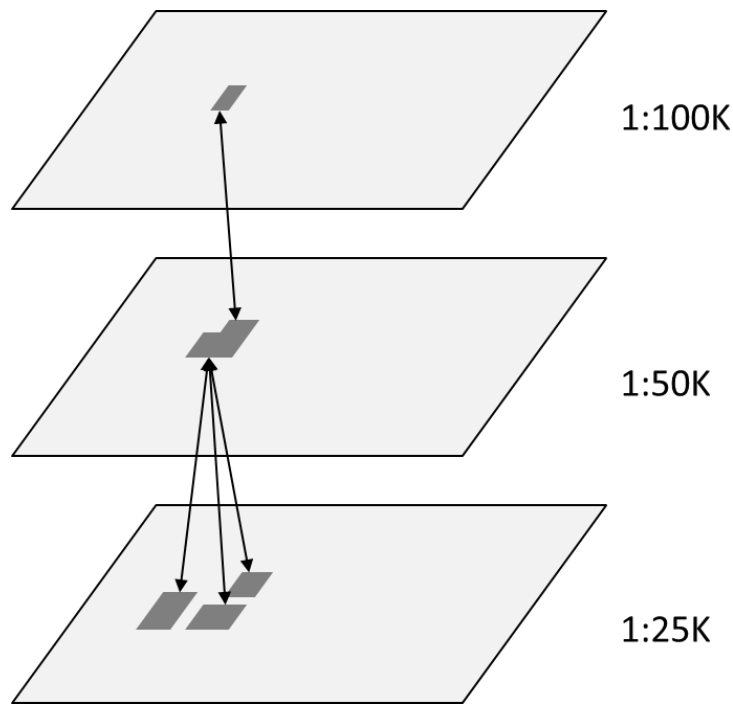


Figure 2.5 A cluster of buildings at three different scales (based on Hampe et al. (2003))

Figure 2.5 depicts a building cluster represented at three different scales in an MRDB. As the scale changes from 1:25K to 1:50K the buildings are amalgamated into a single feature. This represents a *semantic* as well as a *geometric* change. As the scale changes to 1:100K the buildings are still represented by a single object but at a reduced size and simplified shape. MRDB are used by some National Mapping Agencies to manage the process of maintaining multiple map products, each at different scales (Regnauld et al., 2013). For on-demand mapping, Bernier and Bedard (2007) suggest a hybrid solution where, if the data can be generalised automatically at sufficient speed then it should be, otherwise the data should be extracted from an MRDB at the nearest scale to the target scale. Cecconi et al. (2002) also offer a hybrid model but work on the assumption that it is easier to generalise a dataset from a level of detail, provided by the MRDB, close to the required scale than it is from a single highly detailed scale. For each feature type, they seek to define which operators are applicable over a range of scales. They also make the distinction between “on-the-fly” mapping where low processing time is the main motivation and “on-demand” mapping where the cartographic quality of the output is the primary concern.

The effectiveness of an MRDB approach will be determined by the number of layers available in the MRDB. MRDB are not easy to build; they can only be constructed by matching existing datasets or by generalising from a single detailed dataset (Sarjakoski, 2007; Dunkars, 2004). Another problem with this approach is that if the inclusion of user-supplied data is a requirement of on-demand mapping then any such data would have to be integrated with the

data from the MRDB. It is, however, possible to express the *horizontal* relations between features at the same scale, such as that between a building and an access road (Bobzien et al., 2008).

An example of a map created for a specific purpose is the *destination map*, where the aim is to aid navigation from anywhere in a region to a particular point. The web-based generation of destination maps by Kopf et al. (2010) provides an example of automatic generalisation at different scales in real-time. Although the method works at arbitrary scales it has been designed to generalise only two pre-defined feature types, the road and hydrography networks, using a pre-defined workflow.

Sabo et al. (2008) seek to encapsulate generalisation knowledge within “self-generalising objects”, which represent the features that have to be mapped, in a technique similar to the agent-based approach described earlier. They stress that on-demand mapping should be a completely automatic process but do not address how the knowledge required to generalise user-supplied data can be captured.

The “vario-scale” approach (van Oosterom & Meijers, 2011) aims to generalise topographic data to any arbitrary scale. This is achieved by adding a third, scale-dependent, dimension to a 2D representation of the data and then taking a slice through the structure to derive a map at a particular scale. The approach has been used to apply a limited number of generalisation operators to features such as roads, rivers and farmland. However, the authors concede that the approach needs to be extended to include more semantics and context awareness (Meijers et al., 2012).

Corcoran et al. (2011) take a similarly “geometric”¹ approach to on-demand mapping. To generate a map at the required target scale and coverage they use the concept of *geometric coherence*; that is, their model assumes that a requested map has similar geometric data as the previously requested map and that the new map can be generated simply by adding or subtracting from the previous map, thus saving time. However, this approach may be limited to mapping for real-time navigation where geometric coherence is more likely to occur.

Foerster et al. (2012) consider using user profiles to guide the automatic generation of base maps to support thematic data. The profiles describe the topological relationship between the thematic data and the base data on the assumption that it is necessary to preserve the relationship. The relationship between thematic and base data is also a concern of the

¹ All generalisation is ultimately geometric but the approaches in this category take a purely geometric approach.

approach described by Balley et al. (2012) which aims to generate maps with just sufficient information to meet user requirements. They use a rule-base to match “mapped concepts” or features to a template “global” plan.

The different approaches are listed in Table 2.1 and an attempt has been made to classify the type of knowledge that each approach used, considering that, as discussed earlier, it is not always easy to distinguish between procedural and declarative knowledge.

Approach	Priority	Knowledge representation	Reference
MRDB <i>then</i> on-demand generalisation	On-the-fly and on-demand mapping	Declarative knowledge in MRDB	Cecconi et al. (2002)
MRDB <i>or</i> on-demand generalisation	On-the-fly mapping	Declarative knowledge in MRDB	Bernier and Bedard (2007)
Self-generalising objects	On-the-fly mapping	Procedural knowledge in the object definitions	Sabo et al. (2008)
Vario-scale	Multi-scale representation	Procedural knowledge in the slicing algorithm	van Oosterom and Meijers (2011)
Geometric coherence	On-the-fly mapping	Procedural knowledge and declarative knowledge in current state of map	Corcoran et al. (2011)
Plans	Integrating user data	Procedural knowledge in generalisation plans	Balley et al. (2012)
User profiles	Respecting user requirements	Declarative knowledge in the profiles	Foerster et al. (2012)
Destination map workflow	On-the-fly and on-demand	Procedural knowledge in the workflow and algorithms	Kopf et al. (2010)

Table 2.1 Approaches to on-demand mapping

The definition of on-demand mapping by Cecconi (2003) can be modified to stress the importance of automation: the *automatic* creation of a cartographic product upon a user request appropriate to its scale and purpose. The omission of a requirement for real-time, or on-the-fly, mapping from the definition simplifies the problem.

The approach of Balley et al. (2012) is based on the *semantic framework* for on-demand mapping proposed by Regnauld (2007) where knowledge related to all of the components of a future on-demand system, including user requirements, source data, and generalisation

operators and algorithms, would be formalised. This would allow for the sharing of generalisation knowledge and its re-use when applied to different data sources.

2.7 Taxonomies of generalisation operators

The *operator* is a key concept in generalisation (Regnauld & McMaster, 2007) and there have been a number of attempts to classify and define them. Roth et al. (2011) identified 13 classifications prior to their own, with the earliest from 1963. Roth et al. (2011) use the term *typology*, Foerster et al. (2007a) use the term *classification*. There are subtle differences between such terms (Gruninger et al., 2008; van Rees, 2003) but the term *taxonomy* will be used in this discussion.

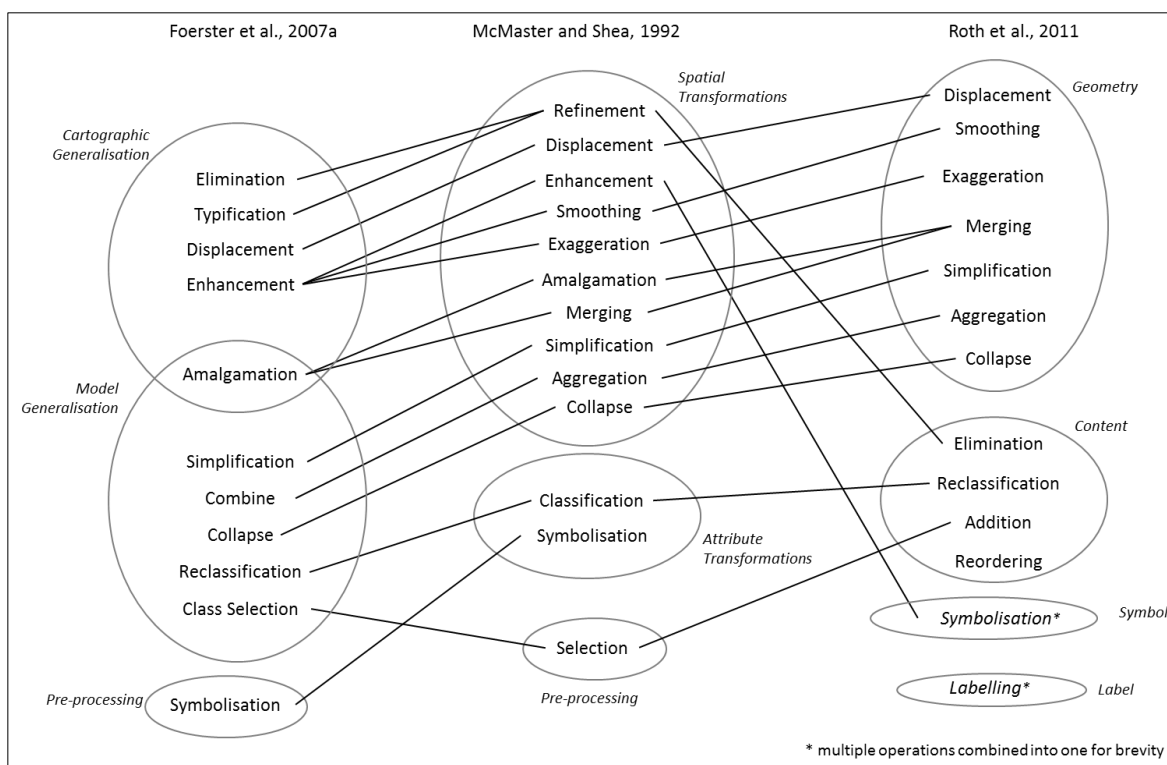


Figure 2.6 Comparison of three generalisation operator taxonomies

The taxonomy of Roth et al. (2011) and that of Foerster et al. (2007a) and the much cited taxonomy of McMaster and Shea (1992) will be compared in an attempt to highlight some of the problems and issues in defining consistent, shared generalisation knowledge (Figure 2.6). The three taxonomies are informal representations of declarative knowledge.

The three classifications suffer from *synonymy*, the application of different terms to the same concept. For example, to describe the process of grouping a set of point features into a single area feature, Foerster et al. (2007a) use the term *Combine* where McMaster and Shea (1992) use the term *Aggregation*. The term *Classification* as employed by McMaster and Shea

(1992) has the same definition as *Reclassification* as used by the other two taxonomies. Synonymous terms are linked by single lines in Figure 2.6.

Polysemy, the use of the same term for different concepts, also occurs. Roth et al. (2011), for example, define *Merging* as the replacement of a group of features with a single feature of the same dimensionality, whereas in the definition of McMaster and Shea (1992), *Merging* only applies to line features.

There are also differences in the granularity of the operators; Foerster et al. (2007a) define an *Enhancement* operator that matches what McMaster and Shea (1992) define in three separate operators; *Smoothing*, *Enhancement* and *Exaggeration*. Such cases are highlighted using multiple lines (Figure 2.6).

There is also disagreement over what operators can actually be considered as generalisation operators. For instance, McMaster and Shea (1992) do not regard the *Selection* of features as a generalisation process but rather a pre-processing step. Foerster et al. (2007a) regard *Symbolisation* in the same manner. Finally, although all three classifications have two tiers, the top-level categories vary in name and meaning.

Kavouras and Kokla (2008) list a number of possible reasons for “taxonomic diversion” that could explain the differences in the taxonomies. These include *perspective*, where the classification is influenced by the needs of a specific application. The Roth et al. (2011) classification, for example, was designed specifically to support the *ScaleMaster* schematic whereas the Foerster et al. (2007a) classification was devised to provide a consistent classification of operators to support web-based generalisation. *Cultural* differences may also apply. For example, Roth et al. (2011) point out that the division between *Cartographic* and *Model* generalisation operators, employed by Foerster et al. (2007a), is particularly dominant in the European literature.

However, even if an agreed taxonomy and description of operators were devised it would not be sufficient for an automated system. A taxonomy can be regarded as an ontology without semantics (Kavouras & Kokla, 2008). For automated on-demand mapping we require formal, machine-readable descriptions of operators rather than natural language descriptions.

Foerster et al. (2007a) claim their classification of generalisation operators as the first to be based on a *formal* model. They use the ISO General Feature Model and OGC GO-1 Application Objects model to describe the impact of the operators. For example, the Amalgamation operator is defined as “Based on *GM_Object* of the original features of the

same *FeatureType* sharing a certain *SpatialAssociationType* a new *GM_Object* will be generated” (p10), where the terms in italics are defined in the ISO General Feature Model. However, such natural language descriptions are still not machine-readable.

It is possible to derive a more semantically rich classification of *algorithms*, as opposed to operators, based on the section titles of the book by Li (2006). He uses at least four levels to organise the algorithms he details in his book (Figure 2.7). The top level is based on the geometry of the features being transformed. His use of the term *transformation* rather than generalisation is convenient given the debate over which operators can be termed *generalisation* operators. He also makes a distinction between algorithms that operate on individual features and those that operate on groups or sets of features. The third level introduces the operator.

In comparison to the three taxonomies described above it is possible to see how a traversal of this hierarchy could be used as a basis for reasoning about generalisation. For example, if the data was a set of point features that required aggregation, then either the ISODATA or the K-means algorithm could be used. Although the geometry of the source data is easy for a mapping system to determine it would still be necessary to determine automatically that aggregation was the required transformation. One weakness of this accidental taxonomy is that it favours a geometric, rather than semantic approach to generalisation.

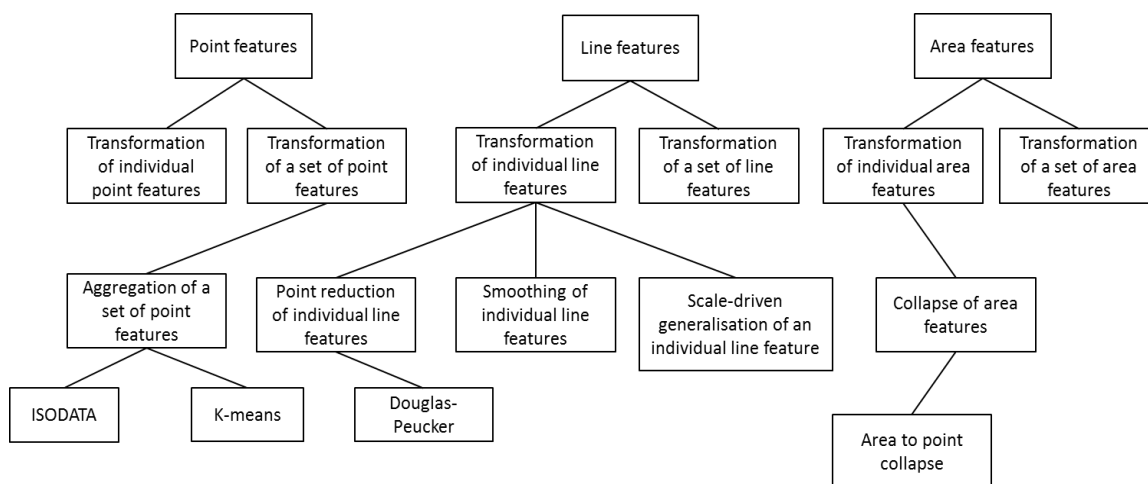


Figure 2.7 A partial taxonomy of generalisation algorithms based on Li (2006)

Although the taxonomies discussed are knowledge representations, they are informal, i.e. not machine-readable. The proposal is, therefore, to formalise generalisation knowledge using an ontology. However, the application of ontologies in cartographic generalisation is not new.

2.8 Ontologies for generalisation

This section looks at applications where geographic knowledge has been made explicit and formalised using ontologies in an attempt to aid generalisation.

Kulik et al. (2005) used information about road class in a line simplification algorithm that assigned geometric and semantic weights to vertices. User profiles were defined which reflected the type of map user. The knowledge was represented simply as quintuples, each element in the quintuple represented a road class; minor roads, major roads, highways etc.

Dutton and Edwardes (2006) used the Web Ontology Language (OWL) to represent the roles of geographic features and semantic and structural relationships between features in a coastal region. Wolf (2009) also proposed the encoding of semantic relationships using ontologies, in particular to influence the aggregation and dimensional collapse of features.

A hierarchical ontology of Geological Time Scales was designed and utilised by Ma et al. (2012) to aid the generalisation of geological maps. Rather than directly influence generalisation, the aim of Luscher et al. (2008) was to use ontologies to aid pattern-recognition in features, in particular to identify terrace houses. Such information could be used to enhance the semantic information in a spatial database, which could then be used to aid decisions on representation.

Iosifescu-Enescu and Hurni (2007) would classify these ontologies as *spatial*, defining the semantics of features, as opposed to *cartographic* ontologies which contain “operational knowledge” required to build legible maps. Their proposed cartographic ontology contains knowledge in the form of cartographic rules aimed at producing high quality web maps. However, their rules are related to symbolisation and are used to represent cartographic conventions related to use of colour and label display and are not concerned with generalisation.

There is a distinction between using ontologies to *aid* the decision making process and using ontologies and ontological reasoning to *make* decisions (Šaša Bastinos & Krisper, 2013). The work described above mainly falls in the former category. Despite that, they fall in that rare category of ontologies that “go beyond simple taxonomies and have been adopted into scientific workflows” (Janowicz et al., 2012, p7). However, none of these ontologies describe the *process* of generalisation, which is the aim of this research.

2.9 Conclusions

This chapter has examined automatic generalisation from a knowledge representation perspective, which has been lacking since the publication of *Map Generalization: Making Rules for Knowledge Representation* (Buttenfield & McMaster, 1991). Since then there has been research into the *acquisition* of generalisation knowledge (Mustiere, 2005; Kilpeläinen, 2000; Weibel et al., 1995; Chang & McMaster, 1993) but little research that reflects the paradigm shift in artificial intelligence from knowledge engineering as a *transfer* process to a *modelling* process (Studer et al., 1998).

The rule-based and constraints-based approaches for automatic generalisation, described above, have limitations; in particular they lack the flexibility that is required by on-demand mapping. This is partly because much generalisation knowledge is embedded in software, either in ArcGIS workflows or in the configurations of agent-based systems. Encapsulating that generalisation knowledge in an ontology will lead to “smarter data” (Carral et al., 2013) where the business logic is transferred from the application to the data. Formalising both the knowledge of the features to be mapped (the data) and the process of generalisation in an ontology will enable that knowledge to be shared and will also allow for reasoning about the selection of operators and algorithms to create maps on-demand. The following chapter will expand on the advantages of using an ontology and describe how such an ontology might be built.

3 An ontological approach to on-demand mapping

3.1 Introduction

The knowledge required for automatic generalisation is currently encapsulated either *formally* and *implicitly* in generalisation software such as agent-based systems (Taillandier & Taillandier, 2012) or *informally* and *explicitly*, for example, in different generalisation operator taxonomies (Roth et al., 2011; Foerster et al., 2007a; McMaster & Shea, 1992). Encapsulating that knowledge explicitly and formally in an ontology will allow it to be shared, expanded and utilised by an on-demand mapping system. This will also result in “smarter” data where the business logic is no longer held in the application but in the data (Carral et al., 2013).

The aim is to design an ontology specifically to solve the problem of on-demand mapping, which requires the implementation of a model for automatic generalisation to provide context for the ontology (section 3.2). This will provide context and help elicit the *concepts* that are relevant to the domain. It is then necessary to identify the *type* of knowledge we wish to represent and section 3.3 justifies the representation of declarative knowledge over procedural knowledge. Knowledge can be represented in a number of ways, it is therefore necessary to justify the use of an ontology (sections 3.4 and 3.5). The type of ontology has also to be considered (section 3.6) before, finally, selecting a methodology that can be used to guide the building of the ontology (section 3.7). The implementation of the methodology is described in Chapter 5.

3.2 A model for generalisation

Knowledge and knowledge representation are context dependent (Kavouras & Kokla, 2008), and a model of the process of generalisation will provide context, determining how the knowledge is reasoned with. Such a model has been described as a “conceptual framework” (McMaster, 1991) and a “conceptual model” (Sarjakoski, 2007) but the intention is the same; it is necessary to provide a high-level description of the process that does not attempt to specify the details of how generalisation should be done.

For an automatic generalisation system a formal, comprehensive conceptual framework is required (McMaster, 1991) and a number of generalisation models have been developed (Sarjakoski, 2007). Of these, the Brassel and Weibel (1988) model has been cited as the most appropriate for computerisation in general (Li, 2007) and for expert systems in particular (McMaster, 1991). The *Why*, *When* and *How* approach to generalisation (McMaster & Shea, 1992), which can be viewed as an extension of the Brassel and Weibel model (Weibel, 1997),

has the flexibility required for the totally automated approach required for on-demand mapping and is particularly appropriate to a semantic approach to generalisation. Their model (Figure 3.1) seeks to define *why* generalisation is required (for example, legibility), *when* it should be employed (when certain geometric conditions such as congestion occur) and *how* it should be implemented (using generalisation operators such as *amalgamation* and *displacement*).

The model starts by considering the reasons *why* generalisation is required. This research is limited, at first, to maintaining map legibility by *reducing complexity*. The model defines a number of geometric conditions that will define *when* generalisation is required. These conditions can be assessed using a number of measures. The final step is to decide *how* to generalise. This is done by applying spatial and attribute transformations, or *operators*. The model also defines three transformation controls *generalisation operator selection*, *algorithm selection* and *parameter (value) selection*. These are the key tasks of an on-demand mapping system, which has to possess the knowledge required to perform these tasks automatically.

The model was revised and simplified for on-demand mapping (Figure 3.1). The major concepts were retained but, for example, the more esoteric concepts such as Gestalt and abstract measures were removed. Also removed were the *Computational elements* reasons for generalisation since they were not a priority for this research. The distinction between *Spatial* and *Attribute* transformations (Figure 2.6) was also removed as it is unnecessary for machine understanding. Added to the model was a new transformation control, *measure algorithm selection*, which is necessary for the approach to on-demand mapping that was adopted. In addition, some of the more imprecise concepts, such as the “conflict” geometric condition, might require revision.

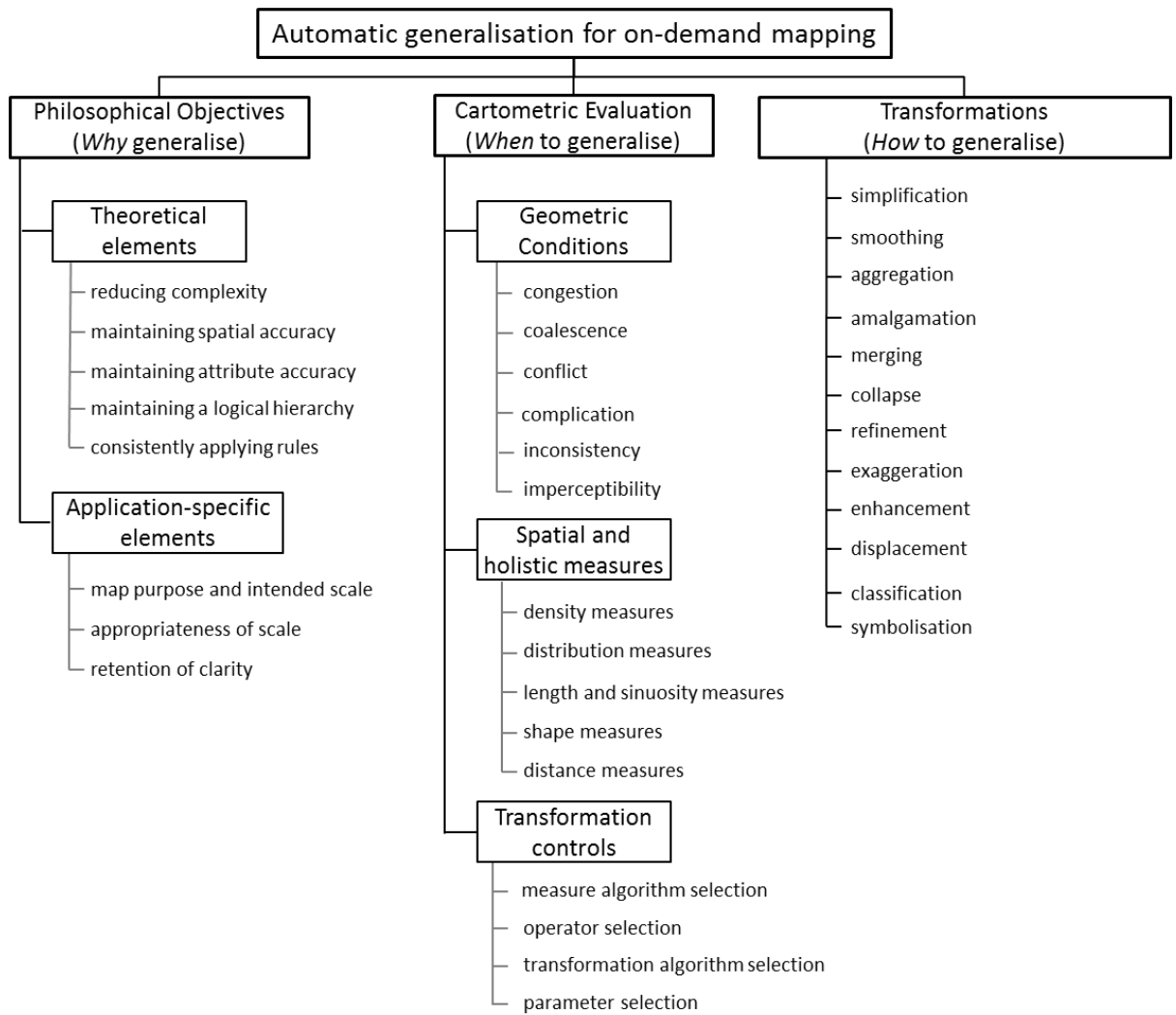


Figure 3.1 The McMaster and Shea generalisation model modified for on-demand mapping

As has been discussed earlier, the knowledge required for *operator*, *algorithm*, and *parameter* selection (the original *Transformation Controls*) is invariably embedded in generalisation software or held by the expert users of those systems. It would be sensible to formalise this knowledge and make it sharable so that the knowledge required to map road accidents, for example, would only need to be defined once and then shared by the different mapping systems.

By framing the process of generalisation as a set of questions the McMaster and Shea model imposes a semantic approach to generalisation in contrast to a mechanistic, unthinking approach such as rule-based systems. The semantic approach is further strengthened if generalisation knowledge is represented as *declarative* knowledge.

3.3 Declarative versus procedural knowledge

Whereas *procedural* knowledge can be characterised as “know-how” knowledge, *declarative* knowledge can be described as “know-what” knowledge (Chen, 2008). While procedural

knowledge describes what to do with the knowledge, the defining characteristic of declarative knowledge is “use-free expression” (Davis et al., 1993).

The main reason for using declarative over procedural knowledge is that it does not prescribe a use and can be “extended, beyond that explicitly represented, by reasoning processes that derive additional knowledge” (Genesereth & Nilsson, 1998, p3). This is done by inference. The advantages of declarative knowledge have not gone unrecognised in the cartographic domain and Kilpeläinen (2000) states “formalisation of knowledge can lead to discovery of new knowledge” (p50). As discussed earlier, it is not always obvious whether particular generalisation knowledge is procedural or declarative. This is the same in other domains, but, Rich and Knight (1991, p395) conclude that it “often turns out that more declarative representations are more flexible than more procedural ones are”.

One disadvantage of declarative knowledge is that its use is slower than directly applying procedural knowledge; efficiency is sacrificed for flexibility (Genesereth & Nilsson, 1998). With declarative statements the user has to decide how to use the knowledge (Davis et al., 1993). More specifically, for an automated system, a program is required that determines how the knowledge is used (Rich & Knight, 1991). The methodology for developing such a system will be discussed in the next chapter.

The intention is to describe the concepts in the *Why*, *When* and *How* model in a declarative manner. That is, to describe sufficiently the concepts of conditions, measures, operators and algorithms so that we can infer procedural knowledge. This will then be used to generalise the mapped geographic features that exhibit any of the *geometric conditions* (Figure 3.1) at any scale. There will still be procedural knowledge contained in the algorithms. But the intention is to use declarative knowledge as much as possible to *select* those algorithms. That knowledge will be represented in an ontology.

3.4 What is an ontology?

The most cited definition of an ontology is that of Gruber (1993): “an *explicit* specification of a conceptualization”. This definition was refined by Borst (1997) as a “*formal* specification of a *shared* conceptualization”. Studer et al. (1998) then merged these two definitions, describing an ontology as a “*formal, explicit* specification of a *shared* conceptualization”.

The key term in these definitions is *conceptualization*, which has been described by Genesereth and Nilsson (1998) as “the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them”. They describe a conceptualization as an “abstract, simplified view of the world”, which is why

conceptualizations are partial, imprecise and conflicting (Kavouras & Kokla, 2008). In effect, a conceptualization is a model of reality. In our case we are trying to create an abstraction (the ontology) of a process (generalisation) that produces an abstraction (a map). Given the complexity of this task it is not surprising that differences, such as those described in section 2.7, arise. An *explicit* specification means that the concepts and their relationships must be explicitly defined (Studer et al., 1998). This is in contrast, for example, to the implicit knowledge held in software.

The specification must be *formalised* in a machine-readable language, which excludes natural language (Guarino et al., 2009). So the natural language descriptions used by the generalisation operator taxonomies (section 2.7) are insufficient. The term “shared”, introduced by Borst (1997), implies that the knowledge encapsulated is consensual, without which the benefits of the ontology are limited. This imposes a limit on the design of the ontology in that, even if it is not constructed by consensus, its concepts and the terms used to describe those concepts must be understood by the ontology’s target audience.

The knowledge representations discussed so far can be classified in terms of their formality and explicitness (Figure 3.2). Cartographers’ knowledge can be classed as implicit and informal or *tacit*. Attempts to make this knowledge explicit have led to a number of generalisation operator taxonomies as described in the previous chapter. Much knowledge has been formalised implicitly in generalisation software but only ontologies are both formal and explicit.

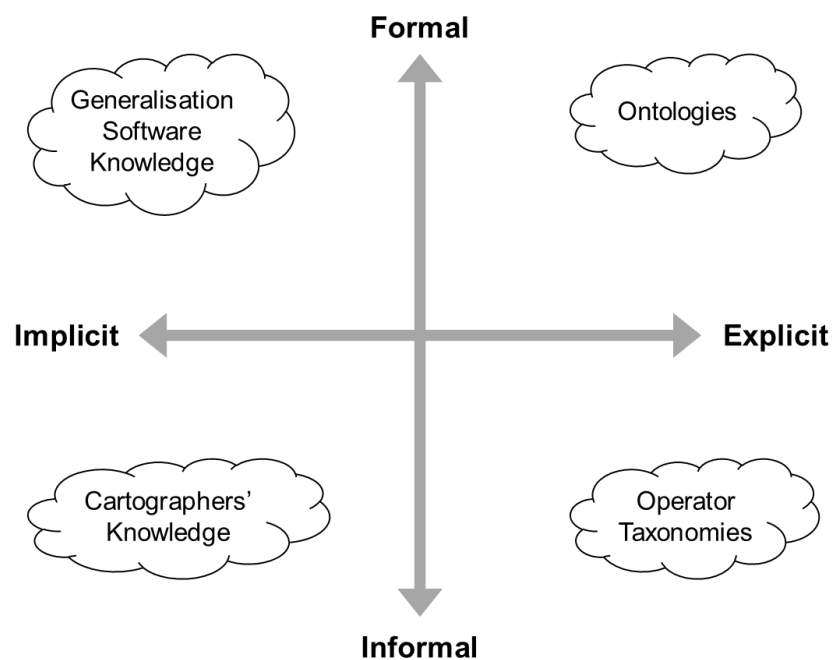


Figure 3.2 Classifying knowledge representations

A key feature of an ontology is the representation of the *relationships* between concepts. As an example, Figure 3.3 gives a view of the possible relationships in the part of the ontology that describes the *FeatureType* concept. The model makes a distinction between *Topographic* (material) and *Thematic* (immaterial) feature types on the assumption that these two feature types will be mapped differently. The *NetworkFeatureType* is defined, in turn, as a sub-class of the *TopographicFeatureType* on the assumption that networks, such as roads and rivers will be managed differently from other topographic features. Most of the relationships between the objects are *hierarchical*, of the form “X is a type of Y”, and an ontology consisting entirely of such subsumption relationships is no more than a taxonomy. However, an ontology can define relationships across classes, for example, the *occursOn* relation between road and accident feature types.

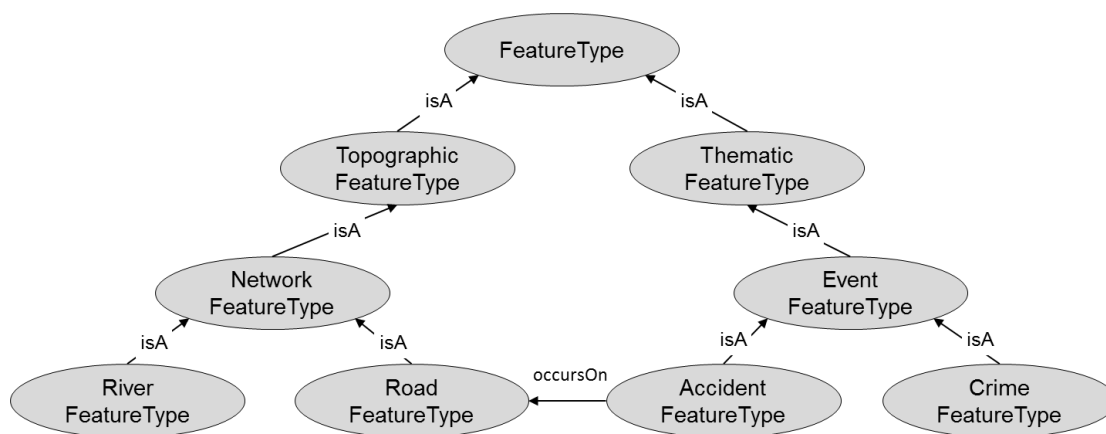


Figure 3.3 Example relationships in an ontology

The structure of the ontology depends on its intended use. If we consider point features we might believe that the generalisation of point events such as crimes and accidents could be done differently from the generalisation of physical point features such as bus stops. It might be sensible to aggregate events such as crimes and accidents to identify hot-spots but not to do the same with bus stops. In this case the definition of an *EventFeatureType* would be useful (Figure 3.3).

3.5 Why use an ontology?

Although accepting that the results of generalisation are the results of manipulating geometric primitives, Mackaness (2007) states that those manipulations need to be based on *context*. That is why, when mapping road accidents, for example, the road network cannot be generalised without taking account of its relationship with the accidents. Context is part of the semantics of a domain, which can be “encapsulated, elucidated, and specified by an ontology” (Kavouras & Kokla, 2008, p10). In the domain of generalisation, the semantic

relationships that govern generalisation must be made explicit and formalised (Wolf, 2009; Dutton & Edwardes, 2006). Over twenty years ago, Nyerges (1991) pointed out that cartographers lacked the means to systematically document the knowledge required for generalisation. The concept of ontologies provides the means.

It is not easy to express the relationships between different feature types, such as those between buildings and access roads, using the hierarchical *object-oriented* data model. However, as we have seen, it is possible to describe such relationships in an ontology since an ontology can not only represent the parent-child relations between objects in a hierarchy but also the relationships that *cross* the hierarchy (Figure 3.3) and even those relationships that cross between different hierarchies.

Using an ontology to describe a domain can lead to intelligent knowledge retrieval (Benjamins et al., 1998). It has been argued that every information system contains an implicit ontology and making it explicit avoids conflicts between the ontological concepts and their implementation (Fonseca et al., 2002). For example, two different algorithm programmers might have different understandings of the concept of the *amalgamation* generalisation operator.

Kavouras and Kokla (2008) assert that the “only intelligent activity justifying the possession of knowledge is reasoning” (p126) and the aim of this research is to use an ontology to aid decision-making, such as selecting a generalisation operator and algorithm for a particular geometric condition. Reasoning with ontologies can be done by inference, which allows us to get “new expressions from old” (Davis et al., 1993). The intention is to encapsulate the concepts of generalisation in an ontology in a sufficiently explicit manner so that it can be used to infer decisions on what operators should be applied. The application of ontologies supports the aim of defining *declarative knowledge* and using it to infer *procedural knowledge*. So rather than explicitly state that geometric condition *X* can be remedied by generalisation operator *Y*, the intention is to describe the characteristics of the condition and operator with sufficient detail to infer that relationship.

A study of the literature reveals that most ontologies are designed to simply formalise, or “fix”, the knowledge in a domain to make it shareable rather than to solve particular problems. There are a number of ontologies that attempt the latter, for example to match students to modules (Kontopoulos et al., 2008) and applicants to jobs (García-Sánchez et al., 2006). In the geospatial domain there are ontologies to aid route finding (Niaraki & Kim, 2009), flood risk assessment (Scheuer et al., 2013), earthquake emergency response (Xu et al.,

2014) and web service selection (Jung et al., 2013). The route finding application of Szwed et al. (2012), which uses an ontology to suggest appropriate algorithms has the most relevance to the this research. However, they use the ontology to encapsulate procedural knowledge, in the form of rules, rather than declarative knowledge. Ontologies designed to solve specific problems seem to be a relatively recent phenomena.

3.6 What type of ontology is required?

A *domain ontology* defines the concepts for a particular domain of interest (Kavouras & Kokla, 2008). Domain ontologies are necessarily comprehensive since they are not coupled with a particular task whereas an *application ontology* will contain concepts that are essential for a particular task and draw on concepts from a relevant domain ontology where necessary (Hart & Dolbear, 2013). An ontology for on-demand mapping using the *How, Why and When* model will contain terms that would not necessarily appear in a general purpose domain ontology for generalisation.

An *application ontology*, or “task” or “problem-solving” ontology, will not be as reusable as an ontology intended to fix the knowledge of an entire domain but will sacrifice scope for the semantic richness required to support useful inference. It may be possible to draw on a generalisation domain ontology such as that proposed by Touya et al. (2010) for some common concepts such as operators, but the intention is to develop an application ontology limited to solving the problem of on-demand mapping. If we accept the assertion by Davis et al. (1993, p29) that “representation and reasoning are inextricably and usefully intertwined” then how we describe generalisation knowledge is dependent on how it is to be used.

3.7 Selecting an ontology design methodology

The discipline that examines the principles, methods and tools for the development of ontologies is known as “ontological engineering” (Sure et al., 2009). Ontological engineering can be seen as a more robust and theoretically sound successor to knowledge engineering (Mizoguchi & Kozaki, 2009). In particular, an ontology engineering methodology provides a guide to the creation of an ontology. A number of methodologies have been developed and it is necessary to select one that is appropriate to the task. This is not however, a straightforward choice since there is no one correct way of designing an ontology (Noy & McGuinness, 2001).

3.7.1 Existing design methodologies

The application of a methodology will provide structure to the process of building an ontology and will ensure best practice (Hart & Dolbear, 2013).

Uschold and King (1995), who were the first to emphasise the necessity of applying a methodology in ontology design (Fernández-López & Gómez-Pérez, 2002), based their methodology on their experiences gained in developing an ontology for enterprise modelling. In the geospatial domain, Torres et al. (2011) developed GEONTO-MET specifically aimed at geographic concepts. The METHONTOLOGY methodology (Fernández-López et al., 1997), in contrast, was intended to provide a generic approach to building ontologies. However, because most methodologies were designed for specific domains or projects, there is a lack of a single dominant methodology (Iqbal et al., 2013).

A literature search failed to find a methodology that advertised itself as specifically targeted at the development of application ontologies so it was necessary to examine a number of methodologies in an effort to find the most suitable. Reviews of available methodologies are a useful starting place. Reviews have been carried out by Fernández-López and Gómez-Pérez (2002), Sure et al. (2009) and more recently by Iqbal et al. (2013) but selecting a methodology based on reviews is not an easy task. Some reviews have restricted domains such as public administration (Stadlhofer et al., 2013) or tend to have a particular perspective; for example, software engineering (Fernández-López & Gómez-Pérez, 2002). Iqbal et al. (2013) define a number of criteria that can be used to aid the selection of an appropriate methodology but their criteria are biased towards reusability and sharing which is not yet a priority for this research, which is currently limited to testing the concept of the applicability of ontological reasoning to the selection of generalisation operators and algorithms.

An alternative starting place in the search for a suitable methodology is to examine the literature for the methodologies used in similar decision-making geospatial projects. However, if formal methodologies have been used to develop geospatial ontologies then their use seems to be rarely documented. For example, Jung et al. (2013) developed eight ontologies for a “geospatial problem solving environment” without describing how the ontologies were designed.

However, despite the diverse origins of ontology design methodologies, a study of four of them reveals areas of overlap (Figure 3.4).

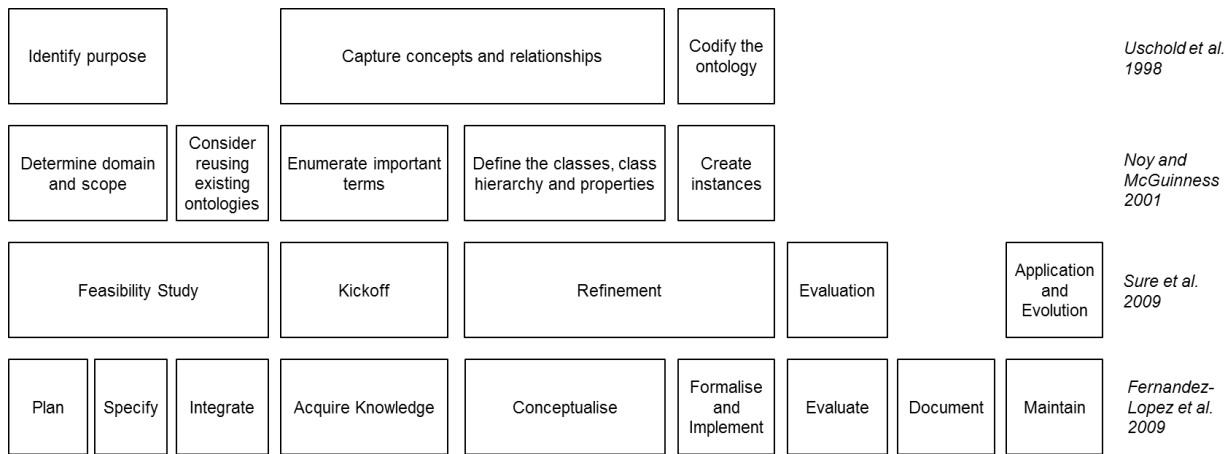


Figure 3.4 Comparable steps in four ontology design methodologies

A common feature among many methodologies is the separation of phases for the informal and formal description of concepts. For example, the Grüninger and Fox (1995) methodology involves the definition of informal and formal competency questions and formal and informal terminologies; Noy and McGuinness (2001) recommend creating an informal list of concepts and their natural language descriptions prior to defining the class hierarchy; and METHONTOLOGY has conceptualisation and formalisation phases (Fernández-López & Gómez-Pérez, 2002). To allow for a less abrupt transition from the informal to the formal it is useful to subdivide the conceptualisation phases further into informal, semi-formal and formal (Figure 3.5) where the semi-formal phase will consist of “mind maps” or, more exactly, directed or undirected graphs (Sure et al., 2009). Some ontology editing tools, such as Protégé (Stanford Center for Biomedical Research, 2014), contain components for viewing ontology structures visually but these are used *after* the ontology has been encoded and are not appropriate for this phase.

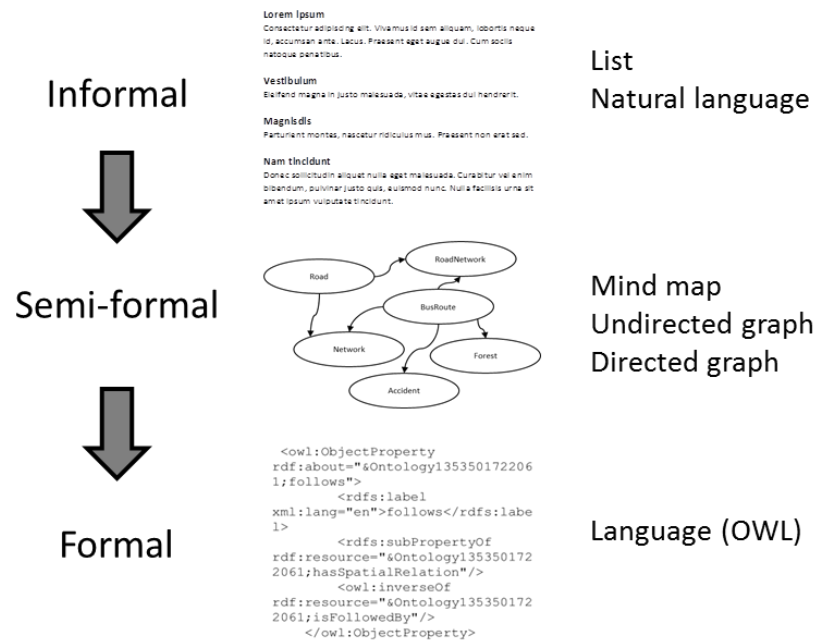


Figure 3.5 Three phase conceptualisation in ontology design

Methodologies such as METHONTOLOGY and UPON (De Nicola et al., 2009) employ a software engineering approach to ontology design. Mirroring the software development approach adds structure and organisation to the process (Saripalle & Demurjian, 2012). This approach has been justified by the inclusion of the term “data” in the IEEE definition of software (Fernández-López & Gómez-Pérez, 2002) but that definition could be regarded as too broad and neither of the two OED definitions of software includes the term “data” (Oxford English Dictionary, 2014c). Fernández-López and Gómez-Pérez (2002) employ the fact that an ontology, by definition, has to be machine-readable as further justification for a software engineering approach but it could be argued that the ontology only becomes machine-readable at the end of the design process.

It might be argued that a software engineering approach is particularly relevant to the development of an *application* ontology and the developers of UPON state that their methodology is not designed to construct generic domain ontologies but user-oriented ontologies, where users include humans and automated systems (De Nicola et al., 2009). However, the development of a knowledge representation should not necessarily mirror that of a software application and even an *application* ontology represents only a part of the whole application. The software engineering approach places too much emphasis on the *management* of the process of designing the ontology rather than the *design process* itself.

There are, however, some aspects of the software development approach that are useful. For example, the UPON methodology advocates the employment of use cases at the start of the

ontology design process since they “drive the exploration of the application area” (De Nicola et al., 2009, p259). Another useful inclusion from software engineering is the iterative approach and a number of methodologies stress the iterative nature of the ontology development process.

One aspect of ontology design that should be considered is *ontology design patterns*, which are small ontologies designed to solve particular, recurring, problems in ontology design (Gangemi & Presutti, 2009). The use of design patterns can lead to the improved efficiency of the design process (Sure et al., 2009) and to a higher quality ontology in terms of its ability to answer competency questions (Gangemi & Presutti, 2009). The Ontology design patterns website² was searched for any relevant patterns but none were found.

The much cited *Ontology Development 101* guide of Noy and McGuinness (2001) is a methodology in all but name. They stress that for an ontology to work effectively it should be designed with a specific task in mind. This is close to the definition of an application ontology. In particular, their approach gives detailed guidance on complex issues relating to the definition of class hierarchies and properties of classes and individuals (or instances) (Iqbal et al., 2013). The guide also gives assistance on making decisions on whether an entity should be represented as a class or an instance. Given that the proposed ontology will not only describe geographic features but intangibles such as events and algorithms, such detailed guidance will prove useful. Indeed lack of sufficient detail has been identified as a common failing among a number of other methodologies (Iqbal et al., 2013).

After studying a number of methodologies including METHONTOLOGY (Fernández-López et al., 1997), GEONTO-MET (Torres et al., 2011), KANGA (Denaux et al., 2011) and the methodologies of Uschold and King (1995) and Grüninger and Fox (1995), it was decided that the approach of Noy and McGuinness (2001) was most suitable for developing an application ontology. However, embracing the assertion of Noy and McGuinness (2001) that there is no single, correct, methodology for designing an ontology, the decision was taken to use their guidelines as a starting point and integrate recommendations from other methodologies where they add value; for example, the semi-formal description of the ontology in the form of directed graphs suggested by Sure et al. (2009) will aid the visualisation of both hierarchical and non-hierarchical relationships between concepts.

² <http://ontologydesignpatterns.org>

The determination of classes and relationships mark the end of semi-formal conceptualisation phase (Figure 3.5). For the next stage, formal conceptualisation, the ontology will need to be implemented in an appropriate machine-readable language.

3.7.2 Formalising the ontology in Web Ontology Language (OWL)

Up to now the discussion of the ontology has been as a high level, abstract, concept and the ontology will need to be formalised using a knowledge representation instrument. Such instruments allow for the examination of the concepts and their relations (Kavouras & Kokla, 2008). According to Rich and Knight (1991) a good knowledge representation system should have *Representation Adequacy* (the ability to represent all of the necessary knowledge), *Inferential Adequacy* (the ability to derive new knowledge from old), *Inferential Efficiency* (the ability to include information that guides the inference process), and *Acquisitional Efficiency* (the ability to acquire information easily).

The last of these, knowledge acquisition, is without the scope of this discussion but the acquisition of procedural generalisation knowledge directly from cartographers (Rieger & Coulson, 1993; Kilpeläinen, 2000) and by using machine-learning techniques (Steiniger et al., 2010; Taillandier, 2007; Mustiere, 2005; Chang & McMaster, 1993) has been well-researched.

Ontologies can be represented in a number of ways using frames, logic, rules and semantic nets (Davis et al., 1993; Shea, 1991). Of these, logic, in particular Description Logics (DL) has high Inferential Adequacy, aiding both the design and deployment of ontologies (Horrocks, 2013).

Ultimately the ontology has to be implemented in a machine-readable language. The Web Ontology Language (OWL) (Antoniou & van Harmelen, 2009) can be used to represent an ontology. OWL has a number of sub-languages of which OWL-DL will be utilised. OWL-DL uses Description Logics to encapsulate the ontology and offers a balance of expressiveness and computational completeness; that is, it guarantees to always return an answer to any query. OWL-Full is more expressive but does not offer computational completeness; it may be possible to query an OWL-Full ontology and not get any answer (Hart & Dolbear, 2013). The Protégé-OWL editor can be used to encode the ontology in OWL-DL (Stanford Center for Biomedical Research, 2014). The representational adequacy of OWL will be tested by the construction of the ontology.

3.7.3 Evaluation

The evaluation of the ontology is seen as a key phase in several ontology development methodologies (Sure et al., 2009; Fernández-López et al., 1997; Grüninger & Fox, 1995). The use of competency questions, which can be used to evaluate whether the ontology meets requirements and act as a justification for the ontology (Grüninger & Fox, 1995), has been adopted by a number of methodologies and is particularly appropriate when developing an application ontology, since a well-defined aim should aid the formation of the competency questions. The optimal extent of the competency questions is disputed; Noy and McGuinness (2001) suggest that they need not be exhaustive whereas Hart and Dolbear (2013) state there should be as many questions as necessary to encompass all usages of the ontology.

The competency questions can be expressed informally in natural language at the start of the process (Noy & McGuinness, 2001) and then formally using the Manchester OWL syntax (Horridge & Patel-Schneider, 2009) once the ontology has been formalised in Protégé. The Manchester OWL syntax provides a simplified method of writing OWL expressions and has been implemented in Protégé. It can be employed using one of the Description Logics reasoners built-in to Protégé such as HermiT (Shearer et al., 2008) or Fact++ (Tsarkov et al., 2006).

A reasoner can also be used to check the ontology's *consistency*, for example ensuring that there are no individuals (instances) in unsatisfiable classes, that is, poorly defined classes that cannot, for logical reasons, have any members. Furthermore, the *OntoClean* methodology (Guarino & Welty, 2002) takes a formal, philosophical approach to evaluating ontologies, as opposed to the application-centric approach of competency questions. OntoClean can be implemented in Protégé.

If the ontology being developed was a general purpose *domain* ontology then the evaluation steps described so far would be sufficient. But since the aim is to develop an application ontology then the ontology can only be evaluated sufficiently by developing a prototype of the system for which the ontology was designed. In this case it meant developing an on-demand mapping system.

One of the fundamental rules of ontology design defined by Noy and McGuinness (2001) is that development is “necessarily an iterative process” and if the evaluation phase revealed inadequacies in the ontology then the ontology will need to be redesigned.

3.7.4 The final methodology

The methodology developed can be seen in Figure 3.6. There is some overlap between phases. For example, evaluation can take place as the ontology is being formalised in Protégé. The method includes a feedback loop in case the Evaluation phase identifies any problems with the design. Whether any redesign occurs at the informal, semi-formal or formal phases depends on the extent of the inadequacies. The more serious the problems the further back in the process it may be necessary to go.

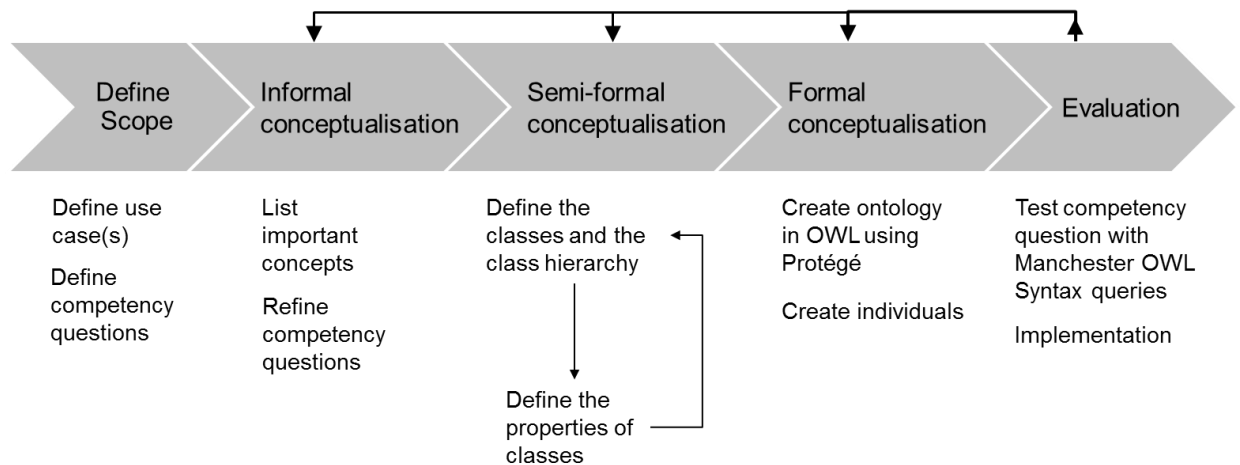


Figure 3.6 Hybrid ontology design methodology

3.8 Conclusions

An assessment of ontology design methodologies revealed the lack of a suitable methodology specifically for the design of an *application* ontology. This was resolved by the development of the novel methodology described above. The application of the methodology to the *How, Why and When* generalisation model of (McMaster & Shea, 1992) is described in Chapter 5. The methodology includes an evaluation phase, which includes the ultimate test of any application ontology: its implementation as part of a software application. The following chapter describes a methodology for implementing a mapping engine that will evaluate and use the ontology.

4 Reasoning with the ontology

4.1 Introduction

The contention of this thesis is that on-demand mapping can be aided by formalising generalisation knowledge in an ontology and the previous chapter discussed a methodology for building that ontology. However, a knowledge representation, such as an ontology, is for “thinking rather than acting... reasoning about the world rather than taking action in it” (Davis et al., 1993, p17). That is, the ontology is not on its own sufficient; a *reasoning agent* is required to make use of the ontology. More specifically, the reasoning agent will utilise the ontology to implement the *How, Why, and When* generalisation model of McMaster and Shea (1992) (Figure 3.1). What is also required is a *mapping engine* that will interact with the user, the geographic data, the reasoning agent and other software components and present the results to the user. These are the main components of the proposed on-demand mapping system.

Furthermore, ontology development is a modelling process, which is typically faulty, and an evaluation of the model is required to ensure its adequacy (Studer et al., 1998). Therefore the implementation of an on-demand mapping system is a necessary stage of the evaluation of the ontology and is part of the ontology design methodology described in Chapter 3.

It could be argued that there was no need to develop an entirely new system and that the ontology could have been integrated with an existing generalisation system. However, this would have provided a less rigorous examination of the ontological approach.

This chapter describes a methodology for building the on-demand mapping system and provides a theoretical foundation for the implementation. The actual implementation of the system is described in Chapter 6.

4.2 Implementing the generalisation model of McMaster and Shea

A flowchart of the process of generalisation based on the adapted McMaster and Shea (1992) model can be seen in Figure 4.1. The process can be divided into two steps, involving the *identification* of a geometric condition in the mapped features and the *remedy* of that condition using a transformation algorithm. The term *generalisation* is not used in the flowchart since there is disagreement amongst domain experts over what processes should be classed as generalisation (section 2.7). Therefore the term *generalisation operator* is replaced by *operator* and the term *generalisation algorithm* is replaced by *transformation algorithm* (to make the distinction from *measure* algorithm). The model describes three transformation

controls which determine the “*How*” of generalisation and are depicted in the flowchart as *Select operator*, *Select transformation algorithm* and *Select parameter values(s)*. A fourth control, *Select measure algorithm*, was added to the model to determine the “*When*” of generalisation (section 3.2). The measure algorithm will implement a *cartometric measure* to determine the existence of a geometric condition. As with the selection of an operator and a transformation algorithm, the selection of a measure algorithm will be supported by the ontology.

The supply of parameter values for transformation algorithms will not be supported by the ontology and the reasons why will be discussed later. The flowchart represents a simplified model of the generalisation process. In particular it applies to a single set of features of the same feature type and it considers only a single geometric condition at any one time. The workflow also depicts generalisation as a strictly linear process.

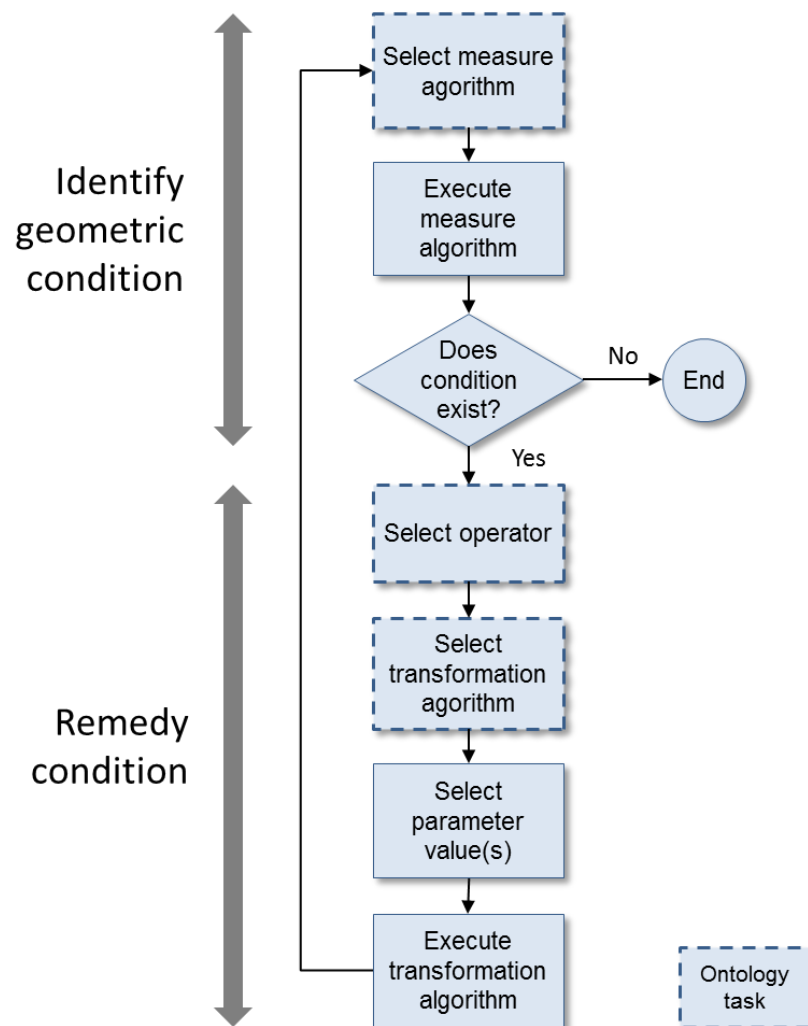


Figure 4.1 The process for generalising a feature collection

The traditional model of a *knowledge-based system* (KBS) is that of an *inference engine* interacting with a *knowledge base* (Studer et al., 1998). However, unlike an *expert system*, the on-demand mapping system is required not only to identify a problem and suggest a solution but also to implement that solution. Therefore only a proportion of the tasks will be carried out using knowledge from the ontology (labelled as *Ontology tasks* in Figure 4.1); other tasks such as the execution of algorithms and the interaction with the user will not be. The component that interacts with the ontology will be called the *reasoning agent* (Figure 4.2). This term is preferred over the more established *inference engine* since the latter implies that the component is driving the on-demand mapping process where, in fact, it is merely a component of a larger system; termed the *Mapping Engine*.

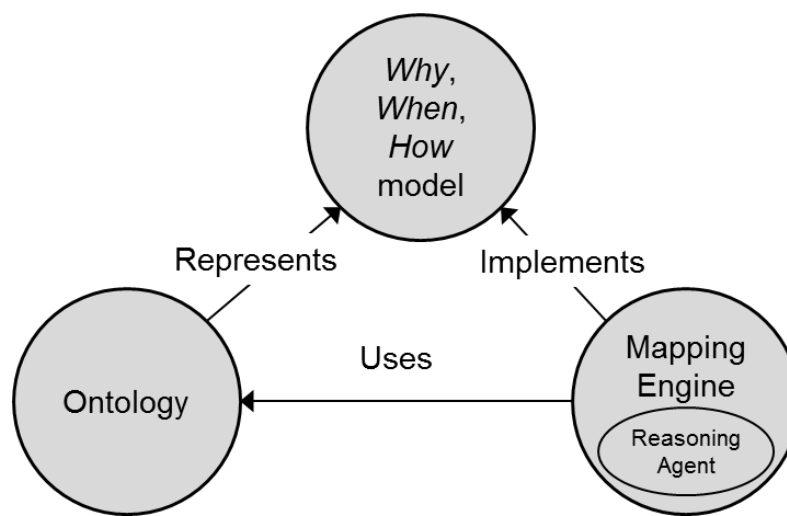


Figure 4.2 The role of the mapping engine

As with the design of the ontology, the design of the system that utilises that ontology (the reasoning agent) will benefit from the structure provided by a suitable methodology. The next section (4.3) discusses that methodology. The selection of transformation parameters will be based on the use of cartometric measures (section 4.4). How these various components will be combined in an on-demand mapping engine is described in section 4.5.

4.3 A methodology for designing the reasoning agent

Knowledge engineering was traditionally seen as a process where human knowledge was transferred to a knowledge-based system (KBS), but it is now regarded as a modelling problem (Studer et al., 1998). To ensure well-formed models, a knowledge engineering methodology is vital (Taboada et al., 2001). The CommonKADS methodology provides a structured approach to the development of knowledge-based systems (Schreiber et al., 2000). The methodology not only describes the type of knowledge required for the application but

also provides a strategy for solving a particular problem (Xavier et al., 2013) and has become the de facto standard for knowledge modelling (Akerkar & Sajja, 2010).

The main feature of the CommonKADS methodology is a collection of models one of which, the *expertise or knowledge model* (Figure 4.3), defines three different types of knowledge that a KBS requires to perform the task. Each knowledge type corresponds to a particular view of the KBS. The expertise model was intended as a knowledge-level description of the system and was developed on the understanding that *control* knowledge, in addition to *domain* knowledge, is an important part of a KBS. The model sub-divides control knowledge into *inference* knowledge and *task* knowledge, where inference knowledge describes how to use domain knowledge with inference and task knowledge describes the tasks that will realise the main goal of the system.

The ontology is contained in the *domain layer* and contains the domain knowledge required to reason about the domain, in this case generalisation. The expertise model does not specify the *type* of domain knowledge that is defined, but in this research the ontology will contain *declarative* rather than *procedural* knowledge. The advantage of using the expertise model is that it indicates how the ontology can be exploited and what further control knowledge is required to develop an on-demand mapping system.

Task Layer <i>Tasks and sub-tasks</i>	Dynamic View	Task Knowledge	Control Knowledge
Inference Layer <i>Problem-solving methods</i>	Functional View	Inference Knowledge	
Domain Layer <i>Ontology</i>	Static View	Domain Knowledge	

Figure 4.3 CommonKADS Expertise model (based on Schreiber et al., 1994 and Studer et al., 1998)

The methodology for building the ontology was described in Chapter 3 but *before* the ontology is built it is necessary to define the control knowledge required by the KBS. This a consequence of the *interaction problem*, which states that domain knowledge is highly dependent on the use to which it is put, which is defined in the control knowledge (Bylander & Chandrasekaran, 1987). It is therefore necessary to describe the knowledge required for the task and inference layers.

4.3.1 Task layer

Task knowledge is held in the task layer, which consists of a hierarchy of tasks and sub-tasks (Figure 4.4) and describes the goal of the system (Schreiber et al., 1994). There is an implied left to right sequencing of the tasks. The leaf-node tasks in the hierarchy, for example *Identify candidate condition* and *Select transformation algorithm* should invoke particular inferences that are described in the inference layer. It could be argued that the *Retrieve dataset characteristics* task does not strictly involve inference but since it involves an interaction with the ontology then it can be classed as such. The tasks in the hierarchy represent a refinement of those in Figure 4.1 but the hierarchy is limited to describing only those tasks that require an interaction with the ontology (domain layer). For example, once the user selects a particular dataset to map, the characteristics of that dataset are extracted from the ontology and used to identify one or more relevant geometric conditions to detect. Using that knowledge, one or more suitable measure algorithms can be selected.

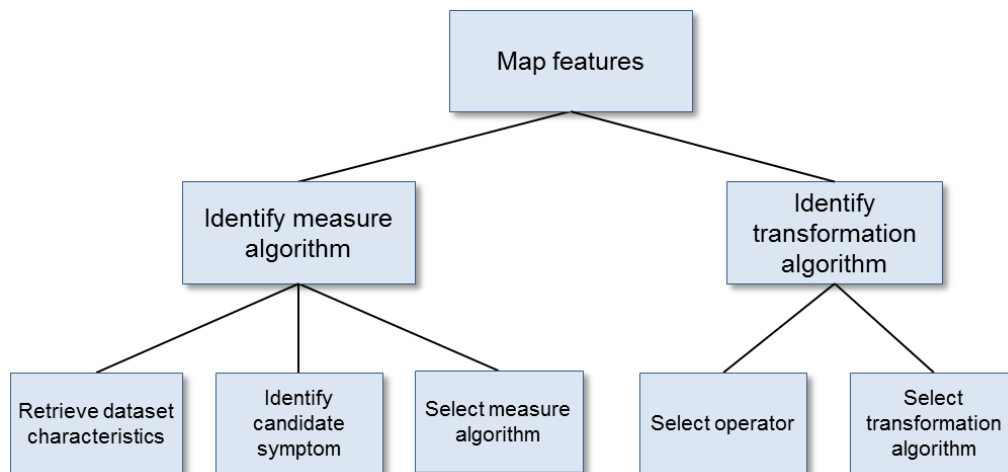


Figure 4.4 Hierarchy of tasks for mapping road accidents at a particular scale

However, the CommonKADS task layer only considers the tasks that are the responsibility of the reasoning agent and not those that are managed by the mapping engine, such as executing algorithms. This disparity is handled by the *system design* model discussed later.

Each task is defined by a *task specification*, which has two parts, the *task definition* and the *task body* (Schreiber et al., 2000). The task definition defines the goal of the task; *what* must be achieved. The task body specifies a procedure for *how* it is to be achieved.

As an example, the specification for a mid-level task from the task hierarchy (Figure 4.4) is displayed in Figure 4.5. The aim of this task is to identify a possible symptom for a set of features, which is an indication of a condition, and then identify a measure for that symptom. The control structure in the task body is a set of functions, where the output of one function

serves as the input to the next. The control structure in this example is very simple, but the model supports more complex structures such as REPEAT-UNTIL loops.

```

task identify-measure-algorithm
  task-definition
    goal: "identify a relevant measure for a set of mapped features";
    input:
      feature-dataset;
    output:
      measure-algorithm;
  task-body
    type: composite;
    sub-tasks: retrieve-dataset-characteristics, identify-candidate-symptom,
      select-measure-algorithm;
    control-structure:
      retrieve-dataset-characteristics(feature-dataset → characteristics)
      identify-candidate-symptom(characteristics → symptom)
      select-measure-algorithm(symptom → measure-algorithm)
end

```

Figure 4.5 Task specification for identifying a measure algorithm

4.3.2 Inference layer – Problem-solving methods

The *inference* layer is the link between the *task* layer and the *domain* layer (the ontology). It describes how the knowledge in the domain layer can be used to satisfy the goals specified in the task layer. In particular it describes how the task definition can be mapped to a task body (Schreiber et al., 1994). The inference layer exploits the notion of a *problem-solving method* (PSM) to specify the reasoning process of the KBS (Studer et al., 1998). A PSM can be viewed as a domain-independent reasoning pattern (Taboada et al., 2001) and the CommonKADS methodology is built on a library of domain-independent PSMs. Domain knowledge (from the ontology) is expressed as a *role* in a PSM which acts as input to an inference step (Role 1 in Figure 4.6). The output of the inference is inferred knowledge (Roles 2 and 3).

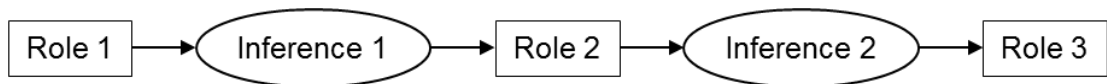


Figure 4.6 Components of a PSM (based on Gómez Pérez and Benjamins, 1999)

The CommonKADS methodology includes a library of reusable PSMs, grouped by category (Schreiber et al., 2000). In this case, the first problem the system needs to solve is to identify any *condition* that the mapped features have, such as *congestion*, which can be done by selecting and executing an appropriate measure algorithm. The library is divided into *analysis*

tasks, such as *diagnosis* and *classification*, and *synthesis tasks* such as *assignment* and *scheduling*. The diagnosis task seems appropriate to selecting a measure algorithm although it could be argued that the problem is one of *classification*.

The PSM should describe the reasoning process in a domain-independent and implementation-independent manner (Gómez Pérez & Benjamins, 1999). For example, the *Heuristic Classification* PSM developed by Clancey (1985) can be used as a diagnostic tool in multiple domains. Figure 4.7 depicts the use of the Heuristic Classification PSM for diagnosing a medical condition. The inference actions and knowledge roles are domain-independent but the labels in italics show what the domain specific knowledge might be.

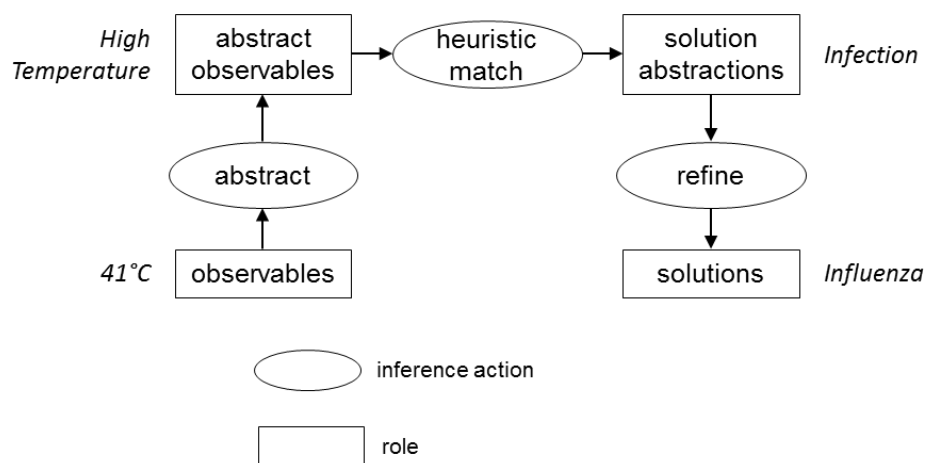


Figure 4.7 Heuristic Classification PSM used for medical diagnosis (based on Studer et al, 1998)

The same PSM can be applied to the diagnosis of a geometric condition in the generalisation model (Figure 4.8). Once the condition has been diagnosed then an appropriate operator can be selected. The medical analogy can be extended if we regard an operator as a *remedy* for a condition. The analogy will be used when it comes to defining the relationships between concepts when the ontology is built.

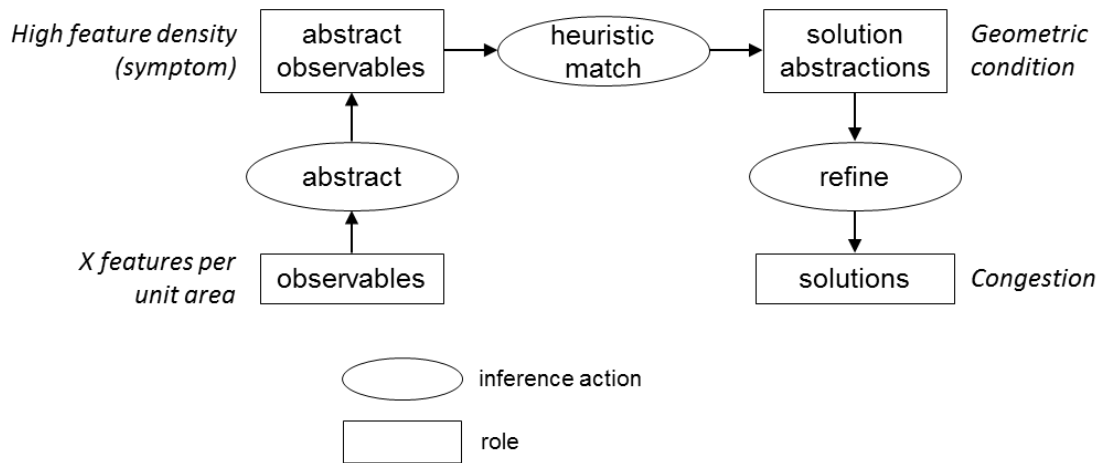


Figure 4.8 Heuristic Classification PSM used for identifying the presence of a geometric condition

However, the Heuristic Classification PSM does not match the tasks defined in the task layer (Figure 4.4) because it assumes that the measurement tool is known in advance; it is equivalent to assuming that the doctor knows how to measure temperature. The on-demand mapping model does not make that assumption. However, the authors of the CommonKADS methodology accept that library of templates can be used as a starting point and then be extended and refined (Schreiber et al., 1994). The Heuristic Classification PSM was therefore modified to allow for the notion that it is first necessary to know what phenomena is to be measured, and how, before measuring it (Figure 4.9). Strictly speaking this structure is not a PSM since it is domain-dependent (but still implementation-independent).

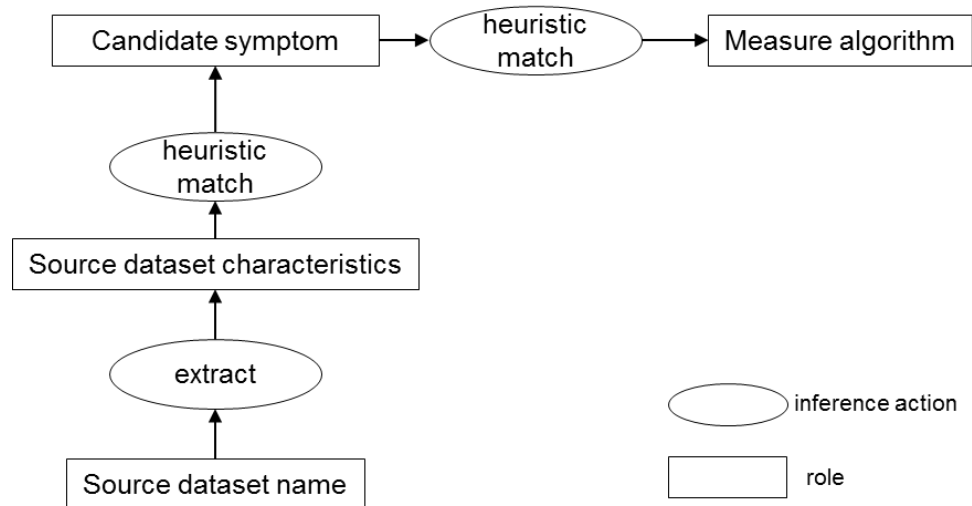


Figure 4.9 PSM for identifying a measure algorithm

Once a geometric condition has been diagnosed, the system needs to suggest a remedy, in particular an appropriate transformation algorithm. As with the selection of a measure algorithm, it is not immediately apparent which of the CommonKADS PSM categories is

appropriate. Based on their descriptions a case could be made for more than one of the analytic tasks including *classification* and *diagnosis*.

An alternative to the CommonKADS library is to examine the literature, and Mustiere (2005) suggested a PSM for generalisation algorithm selection (albeit as part of a machine learning process) and that PSM has been adapted (Figure 4.10). It can be seen how each leaf-node task in the task hierarchy (Figure 4.4) maps to an inference action in the PSM.

The *abstract features* are those features identified as having a particular *condition*, such as congestion, using the diagnosis PSM (Figure 4.8). Selection of an appropriate algorithm requires not only knowledge of the selected operator but also knowledge of the features (the arrows in the PSMs represent inputs and not workflow). This is necessary since many generalisation algorithms have been designed for generalising particular feature types such as roads (Benz & Weibel, 2013; Weiss & Weibel, 2013) and buildings (Steiniger et al., 2010; Regnauld & Revell, 2007). As with the PSM depicted in Figure 4.9, this PSM is domain-dependent but also implementation-independent.

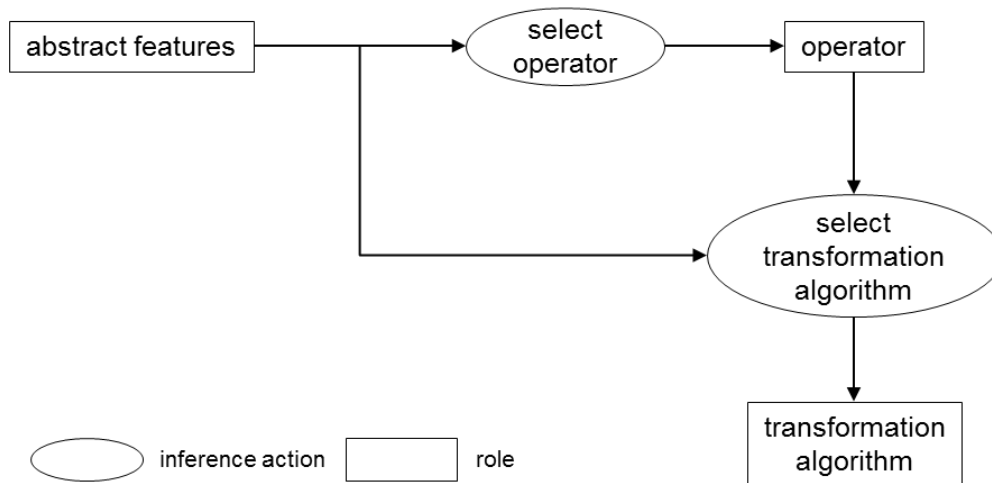


Figure 4.10 PSM for identifying a transformation algorithm (adapted from Mustiere, 2005)

4.3.3 Domain layer

The expertise model does not stipulate an ontology as the source of domain knowledge, but that is what is proposed here. The tasks that determine the PSMs in the inference layer can be used to determine the knowledge that the ontology is required to represent; including concepts such as geometric conditions, mapped features and algorithms. This is why a distinction can be made between an *application* ontology and an all-purpose *domain* ontology. The inference layer can use the ontology to make a number of decisions (Figure 4.1). However the ontology is not used to provide parameter values for the transformation (generalisation) algorithms.

4.4 Parameter selection using the Degree of Generalisation concept

This section describes how the output of the measure algorithms can act as input to the transformation algorithms.

4.4.1 The problem with parameters

Assuming that a system can be developed that will automatically select the appropriate generalisation algorithms then it will be necessary to automatically provide parameter values. For example, the Douglas-Peucker line simplification algorithm (Douglas & Peucker, 1973) has a tolerance parameter that influences the output of the algorithm; a large tolerance will remove more points from a line than a smaller value. For a limited range of target scales and familiar feature types it is possible to use cartographic experience to determine optimum values for parameters but on-demand mapping, by definition, does not allow this.

After reviewing the comparison of line generalisation algorithms by Jenks (1989), McMaster and Shea (1992) concluded that parameter selection has probably a greater impact on the final result than the selection of the generalisation operator and the generalisation algorithm. At the very least, differences in scale will require different parameter values for the same algorithm. Other factors such as feature distribution and the properties of the features may also effect the selection of the optimum values.

The non-expert user cannot be expected to supply parameter values. Parameter names do not provide much guidance; some names such as *smoothing strength* are relatively semantically rich, but others such as *stiffness* and *elasticity* do not even represent geographic concepts and names such as *tolerance* are too vague.

A further barrier to automation is that even those algorithms that implement the same operator do not share the same parameters. For example, line simplification has been implemented by a number of algorithms, including the Douglas and Peucker (1973) algorithm, which has a *minimum distance* parameter, and the Visvalingam and Whyatt (1993) algorithm, which has a *minimum area* parameter. Both parameters do however represent a single concept, tolerance, which influences the amount of simplification, i.e. the number of points removed from the line.

There is also a problem with polysemy, where the same parameter name represents different concepts. For example minimum distance can be used to refer to the distance between a point and a line (Douglas & Peucker, 1973) and the distance between two buildings (Yan et al., 2008). There may also be differences in how that distance is expected to be measured (Qi & Li, 2008). The point feature aggregating algorithms K-means and ISODATA both have a

parameter, K (Figure 4.11), which in both cases represent a number of clusters (Li, 2006). However, for the K-means algorithm, the parameter represents the target number of clusters to be generated whereas for ISODATA the parameter is used only as a stop on the number of iterations and the target is defined by two distance measures ($\sigma_{x,max}$ and $\sigma_{y,max}$).

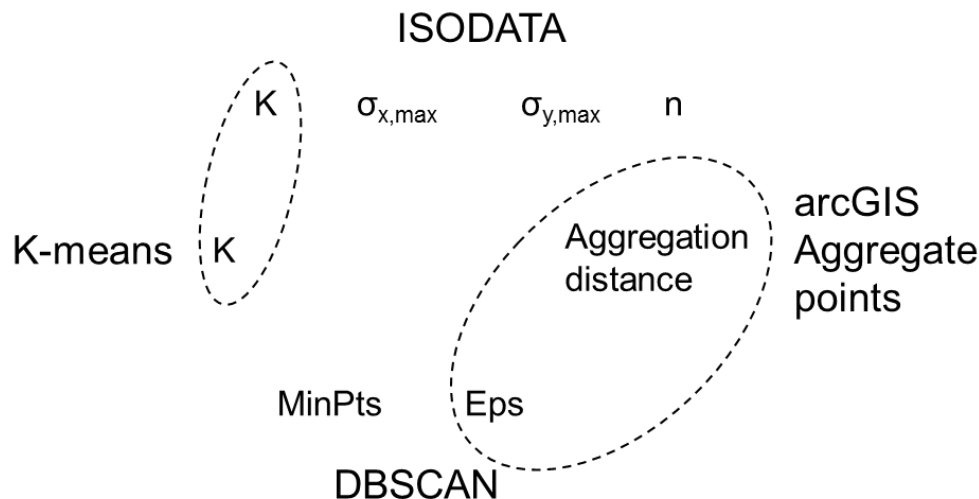


Figure 4.11 Parameters for four point-aggregation algorithms

Synonymy can also be a problem, where the same concepts are represented by two different parameter names. For example, the *Aggregation distance* parameter in the ArcGIS Aggregate points algorithm (ESRI, 2011b) and the *Eps* parameter in the DBSCAN (Ester et al., 1996) algorithm both specify the maximum distance between points that will be aggregated (Figure 4.11).

Algorithms frequently have multiple parameters; some of which are more important than others. Some parameters have an effect on the final output whereas others have an effect on the speed of processing. Some parameters, such as K and n for ISODATA, are optional. Any automatic system should have an awareness of this. Also, parameters may have to be balanced. For example the *snakes* algorithm (Burghardt, 2005), which uses a mechanical analogy and is used for line smoothing, has two parameters, one controlling elasticity and another controlling flexibility (Li, 2006). A successful application of the algorithm depends on selecting the optimum values for the two parameters.

Finally, even when parameter values are set by experts there is anecdotal evidence, from some users of commercial software provided to National Mapping Agencies, that there is some confusion over the effect of some of the parameters, and values are often left at default settings. This may be the result of a lack of documentation of the algorithms.

4.4.2 Possible solutions to the problem of automatic parameterisation

The Radical Law of Selection (Töpfer & Pillewizer, 1966) can be used to determine the number of objects to be retained on a map at a reduced scale, based on the number of source objects and the ratio of the source and target scales. If generalisation was simply a case of determining which features to retain then heuristics could be used to determine parameter values based on the target number of features. However, the Radical Law does not take account of the local distribution of features and does not take into account the transformation of groups of point features into polygons, for example (McMaster & Shea, 1992).

When considering line simplification, Foerster et al. (2007b) defined a *simplification ratio* that could be used by both the Douglas-Peucker and the Visvalingam-Whyatt line simplification algorithms. The value of the ratio is dependent on the user's choice of target scale and is based on the ratio of the number of vertices before and after simplification. The ratio is a variation on The Radical Law, which is also used in a point generalisation algorithm (Yan & Weibel, 2008) as a limit on the number of iterations the algorithm performs. The same algorithm considers the importance value of the source points when deciding to retain a point. By employing the user requirements (target scale) and the source data (number of points and importance values) and by defining neighbours using Voronoi diagrams this algorithm requires no parameters. These concepts could possibly be used to provide parameter values for those generalisation algorithms that do have parameters.

One possible approach is to employ machine-learning methods and neural networks to determine parameter values and then store them in the ontology. Lagrange et al. (2000) employed a machine-learning approach to determine parameter values for the smoothing of individual line features. However, the machine-learning approach is reliant on the existence of a set of reference values (Sester, 2005) and it is likely that this is too complex an approach for on-demand mapping because of the variations in scale and content that on-demand mapping must support.

The method employed in this research is based on the concept of map evaluation. The intention to use the output of the measure algorithms, which determine when generalisation is necessary, to determine the amount, or degree, of generalisation performed.

4.4.3 Map evaluation

The “when to generalise” question in the McMaster and Shea (1992) model is answered using map evaluation techniques or *cartometric measures*, which determine the existence of conditions that affect legibility, such as congestion and imperceptibility. The ontology will

define a relationship between particular measures and particular conditions. For example, a measure of feature density could be used as an indication of *congestion*.

The McMaster and Shea model of identifying the consequences of scale reduction (the geometric conditions) and then identifying the solutions (generalisation operators) has been characterised as a “bottom-up” approach and is particularly appropriate for relatively small changes in scale (Mackaness & Ruas, 2007). The alternative, “top-down” approach involves bypassing the consequences of scale reduction by selecting the features and their representation appropriate to the scale. This approach is considered to be appropriate for large changes of scale; however it requires knowledge of appropriate representations of different feature types at different scales, which will be difficult in on-demand mapping.

Evaluation is carried out by a number of measures that can be categorised as one of three types; those that assess the *amount of information* on the map, those that assess the *spatial distribution* of the map features and those that assess the *complexity* of individual objects (Stigmar & Harrie, 2011). If we consider the amount of information, the simplest measure is to count the number of objects on the map and compare it to a threshold. If the measure exceeds the threshold then it can be concluded that the features have the condition. This is a straightforward task to automate but defining a threshold is difficult since it depends on the feature distribution, symbolisation and the number of feature types. For example, all four representations in Figure 4.12 contain the same number of features with the same distribution yet have differing degrees of legibility since they vary in their symbolisation (Bertin, 1983). An analysis of legibility measures concluded that no single measure could be used to define legibility and a combination of measures is required (Stigmar & Harrie, 2011).

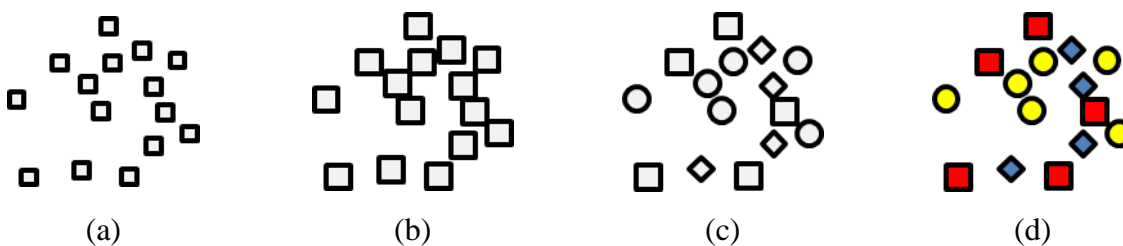


Figure 4.12 Legibility as a function of symbolisation

Despite the difficulties, for any automated mapping system to work it does require a means of self-assessment (Fisher & Mackaness, 1987). If a particular condition has been identified in a set of mapped features then a generalisation operator will be applied and a measure (not necessarily the same one) re-applied to test whether the remedy has worked (Figure 4.1). The process is repeated until the data is free of geometric conditions. The ultimate test of a map’s

legibility is the user's experience. However, the evaluation of the map from the user's point of view is not in the scope of this research.

4.4.4 Calculating a Degree of Generalisation

Not only can the cartometric measures identify whether generalisation is required but they can also be used to determine the transformation algorithm parameter values. If we consider a single set of features of the same type, termed the *source feature collection* (Figure 4.13), then the features the user wishes to map can be termed the *mapped feature collection*. In this example, a measure for feature density (an indication of congestion) has identified two clusters of features, termed *problem feature collections*.

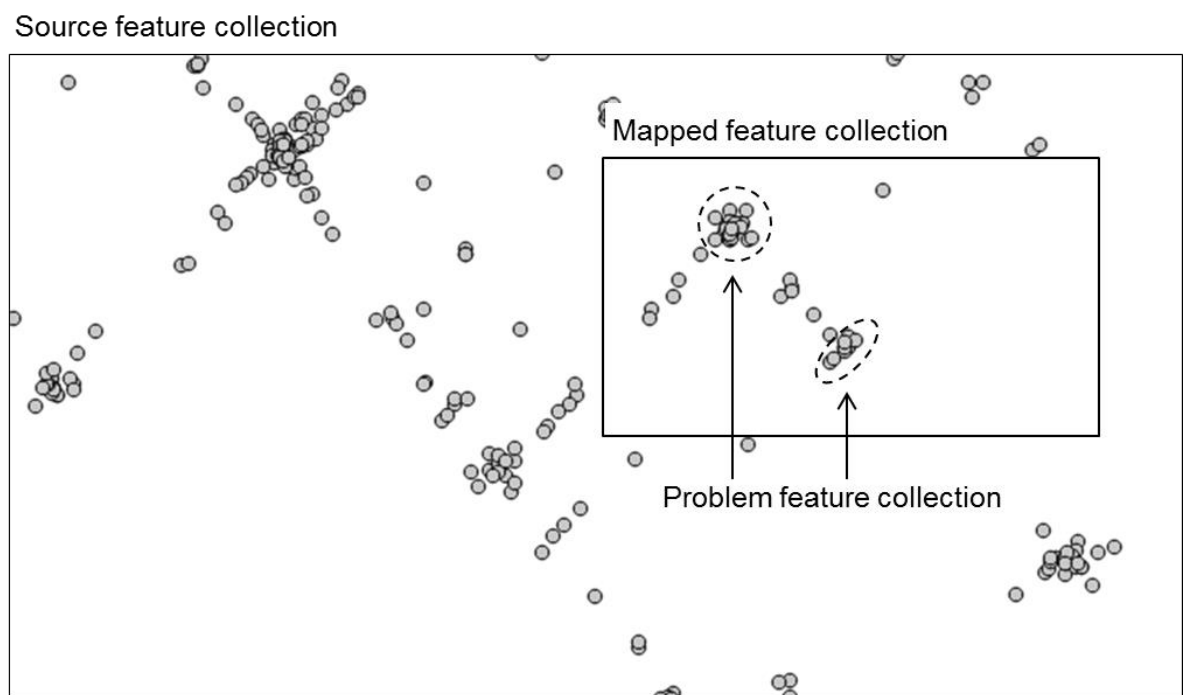


Figure 4.13 Feature collection definitions

Once the problem features have been identified, the concept of a *Degree of Generalisation* (Zhou & Jones, 2003) can be adapted and used as a measure of the “extent” of generalisation that is required. The Degree of Generalisation (DoG) is defined here as the ratio of the number of features in problem feature collections to the number of features in the mapped feature collection (Equation 4.1).

$$\text{Degree of Generalisation} = \frac{\text{number of problem features}}{\text{number of mapped features}}$$

Equation 4.1

Unlike the Radical Law (Töpfer & Pillewizer, 1966) and its derivatives (Foerster et al., 2007b) the DoG concept does not directly use the change of scale to influence generalisation but the *consequences* of the change in scale, which are assessed using cartometric measures. Chapter 6 describes how the DoG can then be used as an input parameter to the transformation algorithms.

The use of the DoG concept does bypass the problem of how to provide parameter values for existing algorithms since it will require a set of generalisation algorithms that have been designed to use the DoG concept. However, the concept of a *translator function* (Touya et al., 2010) that takes the DoG and translates it into algorithm specific parameter values could be applied to existing algorithms.

So far this chapter has described how to perform the selection of a measure algorithm, an operator and a transformation algorithm by using inference and how to provide parameter values using the concept of a DoG derived from measures. The next step is to provide a model of how these components are brought together in an on-demand mapping engine.

4.5 The on-demand mapping system – system design model

The completely automated production of the Netherlands' Kadaster map (section 1.2) was achieved by using many ArcGIS workflows (Stoter et al., 2014). For on-demand mapping a more flexible approach is required. As discussed earlier the system will need to be more than a knowledge-based system since it will need to take the user requirements, make the necessary calls to the reasoning agent, execute the measure and transformation algorithms and present the results to the user. The mapping engine is the equivalent to the derivation engine in the on-demand mapping model of Balley and Regnauld (2011a).

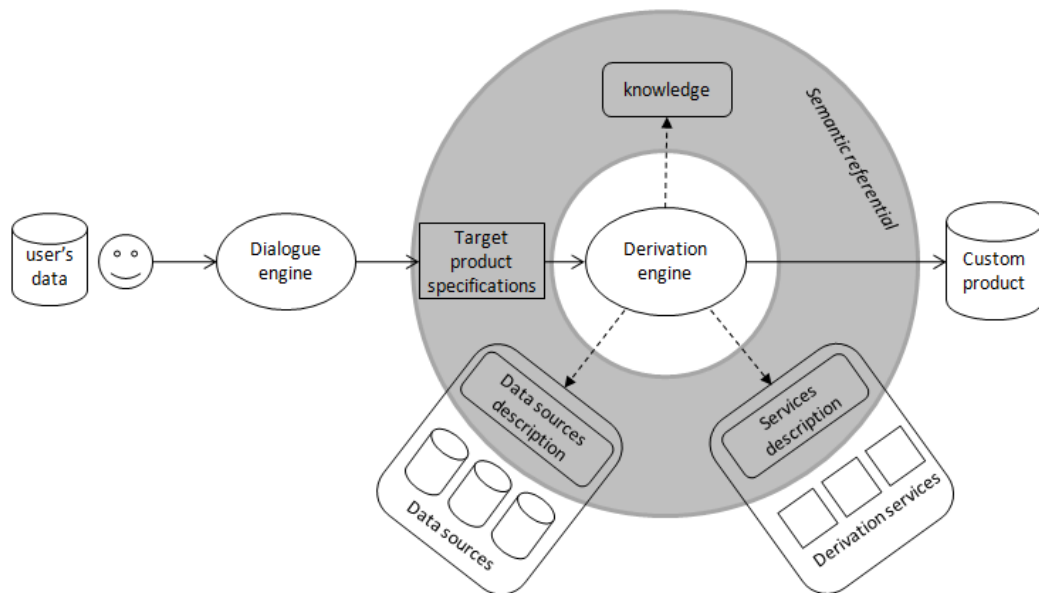


Figure 4.14 On-demand mapping system (Balley and Regnauld, 2011a)

Their model (Figure 4.14) does not specify how the derivation engine produces a custom product from the specification and is not implementation-independent since it is based on web services. However, it provides a useful starting point since it identifies the main tasks and components of an on-demand mapping system.

The CommonKADS methodology provides a *system design* model (Schreiber et al., 2000), which consists of three parts, *application design*, *architecture design*, and *platform design* (Figure 4.15). The last of these, platform design, defines the hardware and software used to implement the system and will be detailed in Chapter 6. The aim of the system design model is to move to platform-dependence as late as possible (Kingston, 1998). Another aim is to preserve the structures of the expertise model as much as possible, such that the expertise model can act as a means of documenting the final system. This has been described as a *structure-preserving* approach (Schreiber et al., 2000). In essence, the system design model provides a means of getting from the expertise model to the implemented system.

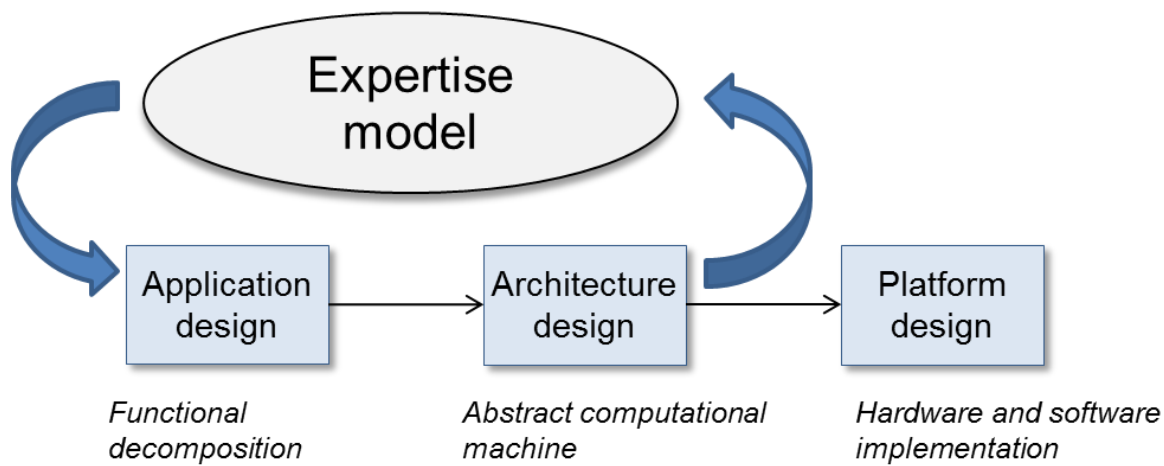


Figure 4.15 CommonKADS system design model (based on Kingston, 1998 and Schreiber et al., 1994)

According to the CommonKADS methodology the task layer only contains inference tasks; however, the on-demand mapping system requires a number of other tasks such as algorithm execution, that do not involve inference and the ontology. The aim is to integrate inference and non-inference tasks in an implementation-independent model that can later be used to design the final system (platform design). One solution to this problem is to employ *transfer functions*.

The CommonKADS methodology allows for *transfer functions* in the task hierarchy, which enable interactions between the *reasoning agent* and other agents such as users (Schreiber et al., 2000). In this case the external agent will be the Mapping Engine. The CommonKADS model describes four types of transfer function: *obtain*, *receive*, *present*, and *provide*. The distinction between *obtain* and *receive* and between *present* and *provide* is based on whether the reasoning agent has the initiative (*obtain* and *present*) or the external agent has the initiative (*receive* and *provide*). In the on-demand mapping system the mapping engine will have the initiative rather than the reasoning agent.

By using the transfer functions as an intermediary between the mapping engine tasks and the inference tasks, an integrated task layer (Figure 4.16) can be developed that combines the tasks of the CommonKADS task layer (Figure 4.4) with those of the mapping engine. The implied sequencing of the task layer in Figure 4.4 is made explicit in the revised version.

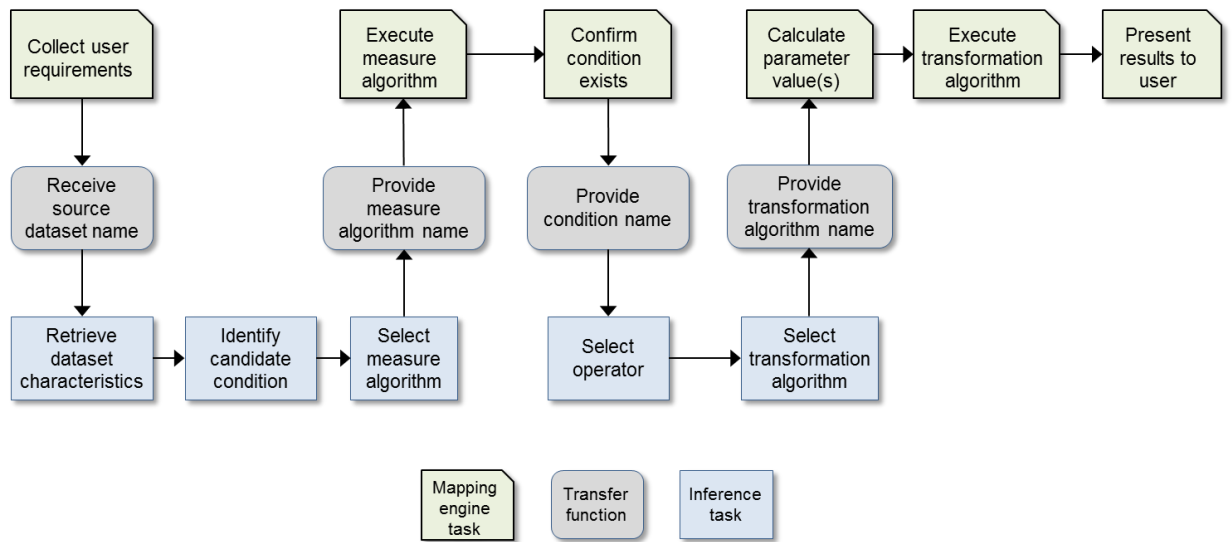


Figure 4.16 Integrated task layer for mapping a single feature set

The integrated task layer is used by the *application design* component of the CommonKADS system design model, where the tasks of the expertise model are mapped to functions of the application (Kingston, 1998). For example, *Identify candidate condition* could be mapped to the function `get_condition(parameter 1, parameter 2...)`. Mapping a single task to a single function is part of the structure preserving approach.

The next stage of the design model is *architecture design* (Figure 4.17) where a computation model able to implement the functions defined in the previous phase is defined (Kingston, 1998). This model should be implementation-independent; an *abstract computational machine* (Schreiber et al., 1994). There is potentially a feedback loop (Figure 4.15) if the architecture design identifies further functionality that is required but was not identified in the expertise model, for example a task that may be split.

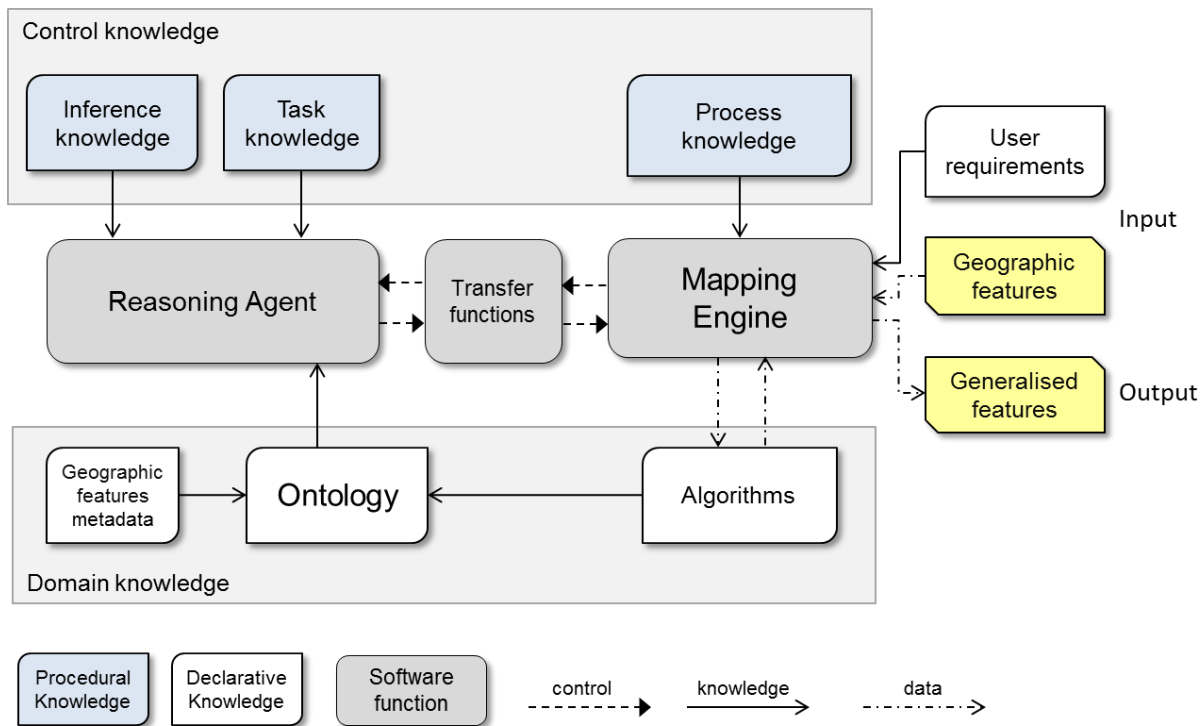


Figure 4.17 On-demand mapping system architecture design

An architecture was designed (Figure 4.17) that not only describes the components of the on-demand mapping system, but also describes, where relevant, the type of knowledge the component represents. The *inference* and *task* knowledge are part of the expertise model whereas the *process* knowledge is that control knowledge that the mapping engine uses to sequence the process of generalisation, from taking the user requirements to applying algorithms to presenting the results, as in Figure 4.16. The algorithms contain *procedural* knowledge but their characteristics, such as what operator they implement and what, if any, feature type they specialise in, will be encapsulated in the ontology as *declarative* knowledge, which will allow them to be automatically selected.

Although the capture of user requirements and their formalisation as a map specification is a necessary component of on-demand mapping (Foerster et al., 2012; Balley & Regnauld, 2011a) it is out of the scope of this research. The user requirements will be limited to selecting the geographic feature types they wish to map and a target scale.

The architecture design will be implemented in hardware and software as the *platform design*, which is described in Chapter 6.

4.6 Conclusions

The contention of this thesis is that much of the *domain knowledge* required for automatic generalisation is encapsulated within proprietary software systems and that to facilitate on-

demand mapping it should be made shareable by formalising it in an ontology. One advantage of the CommonKADS methodology is that it encourages the consideration of the types of knowledge required for a knowledge-based system and where that knowledge is encapsulated. For example, the application methodology has highlighted the fact that *control knowledge* is also required for an on-demand mapping system.

However, the distinction between domain knowledge and control knowledge is not always obvious. It is not clear, for example, whether the formula used to calculate the *Degree of Generalisation* (Equation 4.1) is control knowledge or domain knowledge that should be stored in the ontology. In a case study Schreiber et al. (1994) define an inference step that employs a formula from a knowledge base; the formula is therefore domain knowledge. The more knowledge that is defined as control knowledge, the less the ontology becomes an *application ontology* and the more it becomes an application independent *domain ontology*. Ideally as much knowledge as possible will be described in the ontology since it is shareable and explicit.

There are a number of components of the CommonKADS methodology, such as the *organisational model* (Schreiber et al., 2000), that were not employed. Also, those that were used were not always implemented exactly as defined in the methodology. This was because the methodology is designed for knowledge-based systems but there are tasks in the on-demand mapping system, such as algorithm execution, that do not involve the knowledge base. However, the methodology did prove a useful framework for designing the system.

The concept of building knowledge-based systems with reusable components (PSMs) seems attractive but it has not been widely adopted (Fox, 2011). However, the designers of the CommonKADS state that the library of components can at least provide a starting point for an expertise model (Schreiber et al., 1994) and that proved the case. However, it is the models of CommonKADS, the *Expertise* model and the *System design* model in particular, rather than the library of PSMs that have proved the most useful part of the methodology. In particular it is the decoupling of the knowledge from the implementation, which the methodology supports, that allows for flexibility in the implementation (Wielinga, 2013).

The methodology will be tested in Chapter 6 as will the novel concept of using the Degree of Generalisation concept as a method for automatically parameterising generalisation algorithms. However, before this methodology can be implemented, and the on-demand mapping system built, it is necessary to build the ontology on which it is founded.

5 Building the ontology

5.1 Introduction

A methodology for designing an application ontology was developed in Chapter 3 and will be implemented in this chapter. The content of the ontology will be influenced by the McMaster and Shea (1992) generalisation model and the inference tasks defined using the CommonKADS model (Chapter 4). The content of the ontology will also be influenced by a use case that involves mapping road accidents and the underlying road network at different scales. The methodology is reproduced in Figure 5.1 with numbers that indicate which sections in this chapter describes each phase of the methodology.

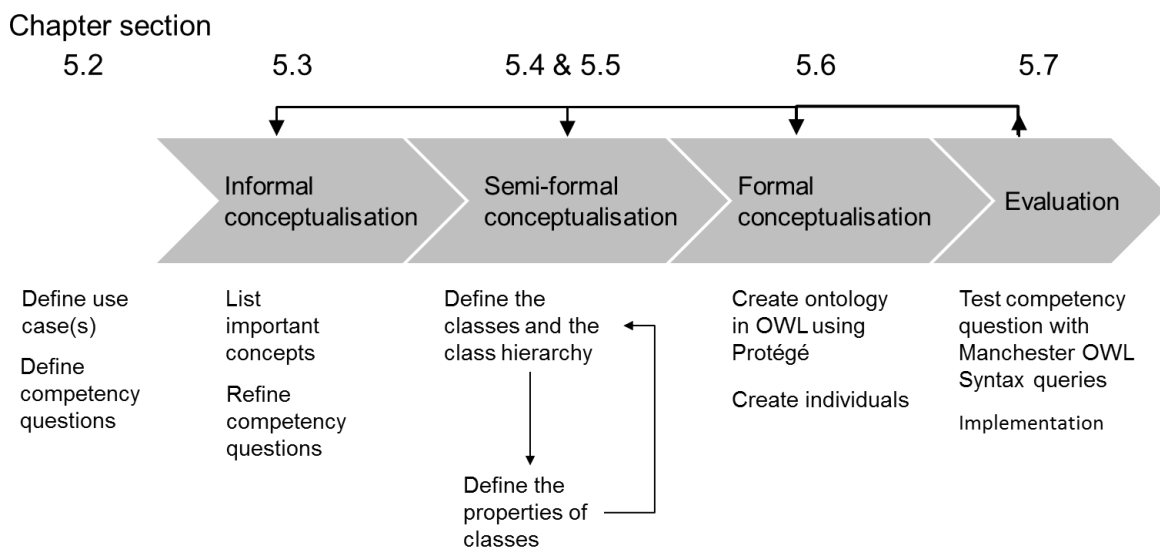


Figure 5.1 Methodology steps with related chapter sections

Generalisation is difficult partly because it is a modelling problem (section 2.2). However, the *representation* of that process in an ontology is also a modelling problem. In effect, we are trying to produce a representation of a process (generalisation) that produces a representation (the map) of both *material entities* (roads) and *immaterial entities* (accidents). To what extent is it possible to represent intangibles such as operators, algorithms, and events in an ontology? According to Davis et al. (1993, p18) it is possible to represent “actions, processes, beliefs, causality and categories” in a knowledge representation such as an ontology. One of the roles of a knowledge representation is to act as a *surrogate*, or substitute, for an entity. Surrogates of real world objects will inevitably be imperfect and will always be simplifications. However, mathematical objects, such as algorithms can be captured exactly since they are “formal objects”. In summary, describing *algorithms* in the

ontology will be relatively easy, describing *geographic features* less so, and describing abstract concepts such as *generalisation* even less so.

5.2 Defining the scope of the ontology

The first phase of the methodology involves defining the scope of the ontology. Since this is an *application ontology* it is necessary to define a use case that will be used to limit that scope. If the ontology can be successfully applied to the use case then it can be tested with further use cases.

5.2.1 Defining the use case

The use case involves mapping road accidents at different scales. This particular use case was chosen as it involves mapping non-topographic, *thematic*, features (the accidents) with *topographic* features (the underlying road network) that share the same geographic space. It fulfils the aim of mapping user-supplied data with data that might be supplied by a National Mapping Agency such as the Ordnance Survey.

The need to vary the feature types mapped according to the user's purpose is well-documented. McMaster and Shea (1992), for example, describe how a farmer and a construction engineer may have different priorities. However, this use case looks at how different users may wish to view the same feature types but at different *scales*. Three types of potential users have been identified; each with a particular reason to map road accidents and each with a particular working scale (Table 5.1).

The road safety expert is interested in identifying particular accident *hot-spots*, areas where a large number of accidents have occurred. This requires a relatively small scale map, showing the city centre for example. A parent wishing to identify a safe walking route to school for a child would require a map at an intermediate, neighbourhood scale. It can be seen (Table 5.1) that the route ACB is of similar length to but safer than ADB. If a particular junction has been identified as an accident hot-spot then the road engineer would need to view the junction in detail. In this example (Table 5.1) it can be seen that the western side of the junction is where most of the accidents occur. In all three cases the road network acts as a context for the accidents; without this context it is difficult to extract useful information from the map.


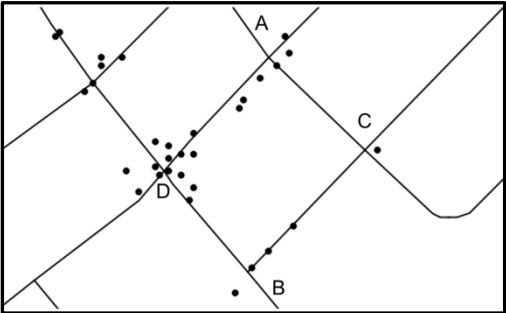
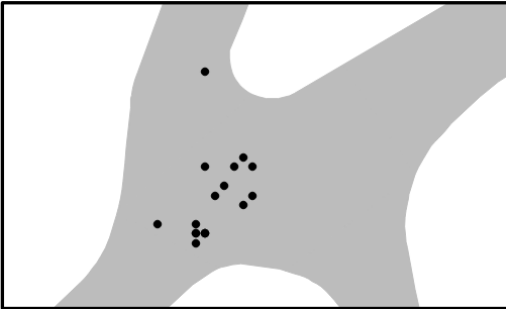
User type	Aim	Relative scale	Example map
Road safety expert	Identify accident “hot-spots” in the city centre	Small	
Parent	Identify a safe walking route to school	Medium	
Road engineer	Identify problem arm at a junction	Large	

Table 5.1 Potential users of an on-demand mapping system

There are reasons other than legibility to generalise geographic data. For example, the *simplification* of line and polygon objects, representing them with fewer points, can be used to reduce data storage and transmission (Bashar et al., 2012). In addition, the aggregation and reclassification of features will allow different analyses of that data (Mackaness, 2007). The generalisation of an urban street network can be used to understand the structure, function and organisation of a city (Jiang & Claramunt, 2004) by allowing the modelling of traffic flows, for example (Jiang & Liu, 2009). The focus of this research, however, is on map legibility and in particular the problem of feature *congestion*. However, it may be that the pursuit of legibility might lead to the greater understanding of the accident data such as the identification of hot-spots.

It is necessary to consider the knowledge that is required to map road accidents at different scales. Firstly, we need to know how to generalise the road accidents, represented initially as

point symbols, at different scales. When mapping *immaterial* entities, such as accidents and crimes, which can share the same geographic space, the user may find some degree of overlapping acceptable. Whereas the overlap, caused by a change in scale, of *material* entities such as topographical features, would not be acceptable. At smaller scales, however, the degree of overlap of accident symbols may be excessive and generalisation is required (Figure 5.2).

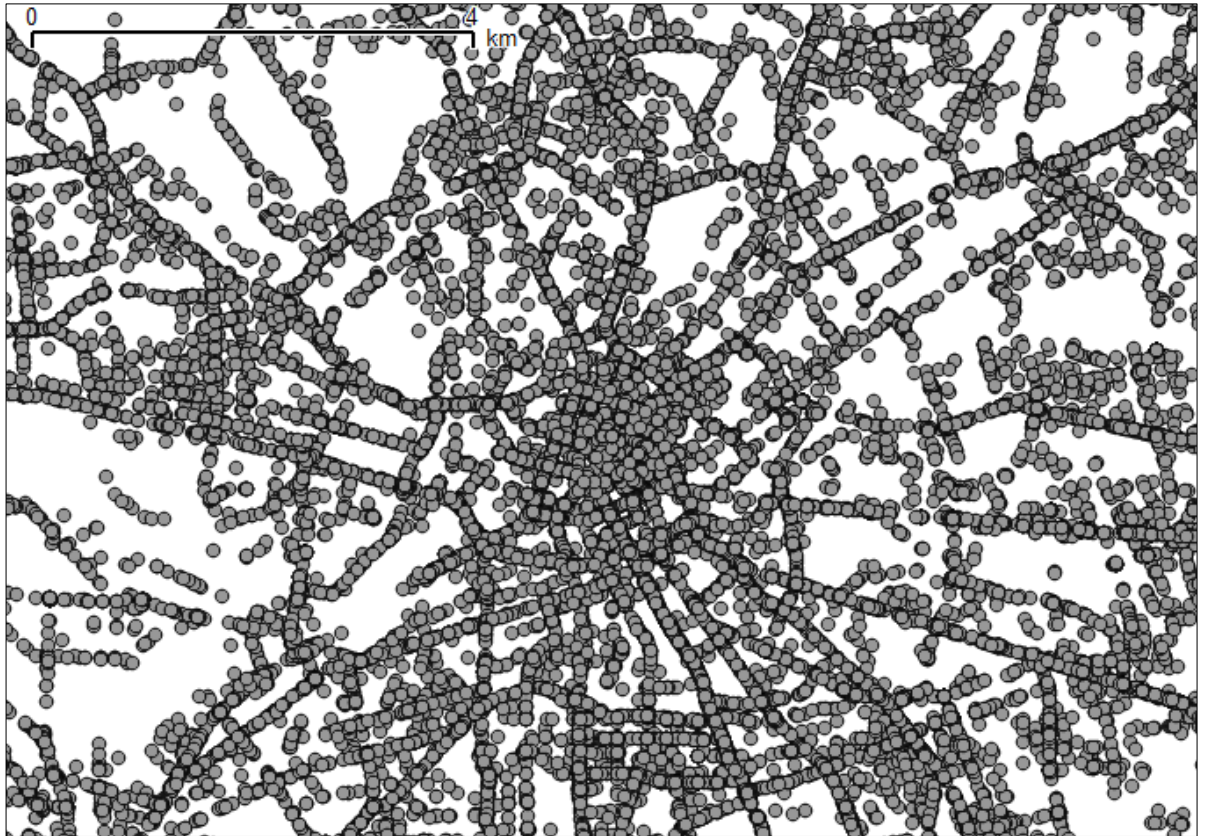


Figure 5.2 Road accidents at 1:60K (data provided by Transport for Greater Manchester)

Congestion occurs as a consequence of scale reduction when too many features need to be represented in the same geographic space (McMaster & Shea, 1992). This phenomenon has also been described as *clutter* and is not always straightforward to define (Stigmar & Harrie, 2011) but for the purpose of this research it is to be regarded as occurring when the feature density is too high. There are a number of ways of resolving high feature density; reducing the number of features, reducing their size, and moving the features apart (Figure 5.3).

Reducing the size of features can be classed as *symbolisation* (Figure 5.3b), which has been defined as “the graphic encoding of a feature on the map page” (Roth et al., 2011, p39). However, symbolisation is not always regarded as a generalisation process, but as a pre-generalisation process (Foerster et al., 2007a). Whether or not symbolisation can be classed as generalisation, it is true that it can be used in many ways to improve map legibility,

including changing the size, shape, colour, transparency and pattern of features (Roth et al., 2011) (Figure 4.12). However, to limit the scope of this research, the decision was made to focus on the fundamental, “spatial” operators (McMaster & Shea, 1992) and ignore symbolisation.

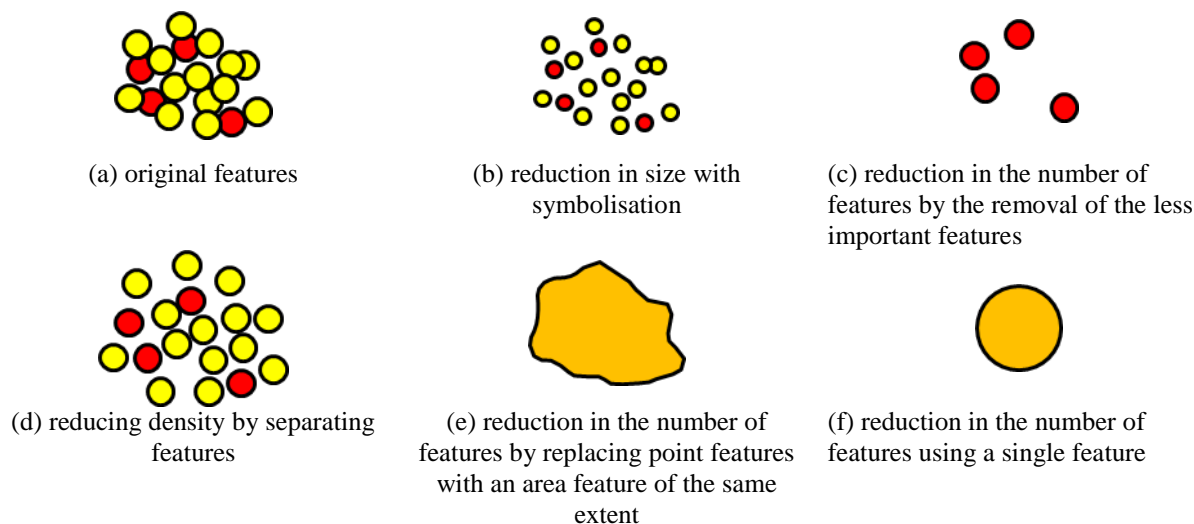


Figure 5.3 Resolving congestion in accident data by reducing feature density³

Of the other possible solutions to congestion, the separating of features (Figure 5.3d) presents problems. According to the McMaster and Shea (1992) model *maintaining spatial accuracy* is one of the goals of generalisation but at a large scale the displacement of features might cause an unacceptable loss of spatial accuracy. At a relatively small scale (such as Figure 5.2) the displacement of features may not be possible due to the lack of vacant map space in which to move the features. This is a particular problem for events, such as accidents, where multiple features may have the same location.

When features are congested then some sort of grouping may be helpful i.e. replacing a cluster of features with a single feature (Figure 5.3e and Figure 5.3f). So a group of clustered accidents would become an accident hot-spot. This can be regarded as an example of *conceptual* generalisation, where new information has been derived; “the *tree* gives way to the *forest*” (Bertin, 1983, p300). However, this operator is relatively invasive (Foerster et al., 2007a) and leads to the loss of individual detail.

An alternative is *structural* generalisation, where the level of conceptualisation remains the same but the distribution is simplified (Bertin, 1983). One way of achieving this would be to

³ This part of the discussion avoids reference to operators, focussing on the solutions and not the means.

reduce the number of features by displaying only the most important (Figure 5.3c). For this to be possible the data requires an attribute that can be used to rank features by importance. In the case of accidents this could be a *severity* value. The on-demand mapping system would need to know whether the dataset possessed such an attribute and this information would be part of the formalised knowledge of the dataset.

If we define a road accident as an event that occurs at a particular point on a road then we can infer that road features provide necessary context when mapping the accidents. Therefore we need to map road features (Table 5.1). How the road network is represented is also important; to examine a road junction in detail then the road features should be represented as area features whereas at smaller scales a line representation is sufficient.

At smaller scales the road network too might require generalisation (Figure 1.1). In this example, the density of features is too high in many places and it is necessary to remove the less important roads segments to reduce congestion (Figure 5.4c).



Figure 5.4 Reducing feature congestion in road segments by reducing feature density

It is also possible to reduce feature density by reducing the size of features with a reduction in dimensionality (Figure 5.4b). This process is generally regarded as generalisation whereas the reduction in size of features using symbolisation (Figure 5.3b) is not.

The generalisation of road networks has been well-researched (Benz & Weibel, 2013; Weiss & Weibel, 2013; Touya, 2010; Chen et al., 2009; Jiang & Harrie, 2004; Thomson & Richardson, 1999) as has the generalisation of point data, although to a lesser extent (Bereuter & Weibel, 2012; Yan & Weibel, 2008). However, the mapping of *both* accidents and the underlying road network presents a particular problem as the relation between roads and accident needs to be represented and respected when the road network is generalised, otherwise context may be lost. As the scale is reduced the standard method to ensure the legibility of the road network is to prune the network by removing the less important roads. However, as can be seen in Figure 5.5, a loss of context for some of the accidents can be the

result. The semantic (and ultimately spatial) relationship between roads and accidents has to be defined in the ontology and respected during generalisation.

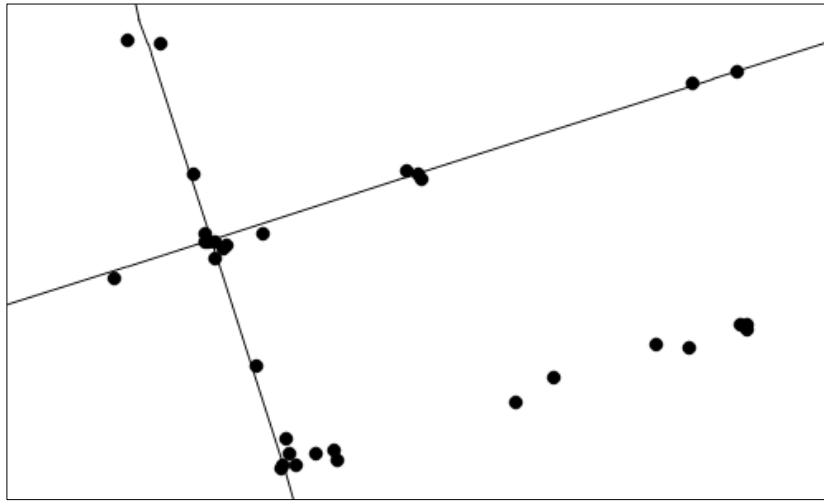


Figure 5.5 Loss of context for the accidents

Some of this knowledge is generic and some is specific to road accidents. Consider another point feature type such as a *crime*, which is also an event and could be generalised in a similar manner to accidents; either by grouping or removing less important features. However, the contextual features would be different; buildings, for example, may be more relevant as a base feature layer.

This section has described some of the knowledge required to map a single feature type, road accidents, with the road network providing context, at different scales. Given the large number of feature types that users may wish to map, just a few of which were listed in at the start of Chapter 1, there is a case for sharing this knowledge amongst mapping systems by using an ontology.

5.2.2 Competency questions

The aim of the ontology is to aid the automatic selection of generalisation algorithms. The domain is *on-demand mapping* in particular and not *cartographic generalisation* in general and the ontology needs to hold only sufficient knowledge to solve the problem of on-demand mapping. In particular, the aim is to encapsulate the knowledge necessary to implement the *Why*, *When*, and *How* model of (McMaster & Shea, 1992). Inference will then be used to determine which generalisation algorithms should be applied in any given situation.

In the second part of the *Define Scope* phase of the methodology the informal *competency questions* (Grüninger & Fox, 1995) can be framed, although at this stage they may be relatively vague. Examples of competency questions are

- “What are the geometric conditions that might affect the accidents when mapped at a smaller scale?”
- “When should the road network be generalised?”
- “How can the road network be generalised?”

Noy and McGuinness (2001) suggest that the competency questions need not be exhaustive and this was the approach adopted here. A sufficient number of questions were framed to ensure that they focussed the design process rather than act as a specification.

In summary, the scope of the ontology is limited initially to the use case. However, some sub-concepts that are *not* necessary for the use case such as additional feature types and operators will be included since they can act as control information. For example, if the only operators that can be selected are those that are applicable to the use case then it is not a sufficient test of the approach.

5.3 Informal conceptualisation

This stage of the methodology matches Step 3 of the Noy and McGuinness (2001) methodology: *enumerate important terms in the ontology*. An ontology encapsulates the *semantics* of the concepts of a domain (Kavouras & Kokla, 2008) and is therefore more than a list of concepts. The enumeration will require descriptions of the concepts and their relationships with other concepts. At this stage no structure needs to be applied to the knowledge.

To guide this process the third of the fundamental rules of Noy and McGuinness (2001) can be employed

“Concepts in the ontology should be close to objects (physical or logical) and relationships in your domain of interest. These are most likely to be nouns (objects) or verbs (relationships) in sentences that describe your domain.” (p4).

Some of the main concepts will be *highlighted* in the following discussion which is based on concepts from the McMaster and Shea (1992) model and the tasks listed in the *Expertise* model (section 4.3.1)⁴.

The *feature collection* (FC) concept will be used to represent a set of features. The *source feature collection* represents the entire dataset, the *mapped feature collection* represent the

⁴ This is an iterative process. The tasks in the Expertise model were revised following the conceptualisation.

features in the user's selected bounding box. Features that exhibit any of the *geometric conditions* in the model, as identified by the *measures*, will be marked as *problem features* (Figure 4.13). Each feature collection is composed of objects of the same *feature type*; *accidents* or *road segments*, for example.

The scope of this research is limited to measures of *legibility* (Stigmar & Harrie, 2011). A *measure* does not directly measure a condition. In this interpretation of the McMaster and Shea model a condition is characterised by one or more *symptoms*, which can be identified by the measures. For example, *congestion* can be characterised by *high feature density* which can be measured by a *density measure*.

Measures will be implemented by *measure algorithms*, the selection of which will be based on the geometry of the mapped feature collection. For example, feature density will be a relevant measure of congestion for features with any *geometry*; feature complexity will be a relevant measure of imperceptibility applicable to *line* and *area* geometries only. The measure algorithm will return a set of problem feature collections identifying features in the mapped FC with a particular condition.

The next stage is to identify the *operators* that will *remedy* the *symptom* and by inference the condition. An operator is *implemented by a transformation algorithm*. The term *generalisation algorithm* is avoided since there is debate over which cartographic operators can be classed as generalisation (Foerster et al., 2007a; McMaster & Shea, 1992; Roth et al., 2011).

Some of the concepts highlighted above are described in Table 5.2. If another concept is included in the description then it is underlined.

Concept	Description
Cartographic generalisation	<i>"The abstraction, reduction, and simplification of <u>features</u> so that a map is clear and uncluttered at a given scale"</i> (Sommer & Wade, 2006). This definition is used in preference to the frequently cited International Cartographic Association (1973) definition: <i>"the selection and simplified representation of detail appropriate to scale and/or purpose of a map"</i> since it better represents the aims of the use case.
Congestion	The <u>geometric condition</u> where <i>"too many geographic <u>features</u> need to be represented in a limited physical space on the map"</i> (McMaster & Shea, 1992).
Geometric Condition	Conditions in the mapped <u>features</u> that are caused by a reduction in scale and used to determine the need for <u>generalisation</u> (McMaster & Shea, 1992). For example, <u>congestion</u> .
Event	An incidence or occurrence (Oxford English Dictionary, 2014b); an immaterial object that can share the same space as a <u>topographic feature</u> or another event.
Feature	A mapped object. Can be <i>material</i> (topographic) or <i>immaterial</i> (such as an event). Features can be grouped by <u>feature type</u> . E.g. Buildings.
Feature collection	A set of <u>features</u> all of the same <u>feature type</u> .
Feature type	A class of features e.g. buildings, rivers.
Geometry	<i>"The measures and properties of points, lines, and surfaces. In a GIS, geometry is used to represent the spatial component of geographic <u>features</u>"</i> (ESRI, 2014).
Operator	<i>"Abstract or generic description of the type of modification that can be applied when changing scale"</i> (Roth et al., 2011). An abstract function that transforms geographic data. An operator is implemented by one or more <u>algorithms</u> .
Road accident	Point event <u>feature type</u> . Takes place on a <u>road segment</u> .
Road segment	Section of road between two junctions (nodes). A topographic <u>feature type</u> . Part of a <u>network</u> .
Symptom	<i>"A phenomenon or circumstance accompanying some <u>condition</u> ... and serving as evidence of it"</i> (Oxford English Dictionary, 2014a)
Measure algorithm	A procedure for measuring a particular <u>symptom</u> .
Transformation algorithm	A procedure for implementing a particular <u>operator</u> . Some transformation algorithms specialise in particular <u>feature types</u> . Normally termed a <i>generalisation</i> algorithm. Implemented in computer code.

Table 5.2 Example concept descriptions

5.3.1 New knowledge from old

It is possible to encapsulate the relationships between operators and conditions as rules within the ontology. Szwed et al. (2012), for example, used an ontology to store rules encoded in the

Semantic Web Rule Language (Horrocks et al., 2004) to aid the selection of algorithms for route-finding. However, there are a number of problems with a rule-based approach to generalisation which have been well-documented (Harrie & Weibel, 2007; Armstrong, 1991; Beard, 1991) and discussed in section 2.4. A particular problem is the large number of rules required for a complex process such as generalisation. We also cannot simply state that operator *X* resolves geometric condition *Y*. This is, in effect, a rule

IF geometric_condition = *Y* THEN apply operator *X*

The aim, therefore, is to formalise the characteristics of the operators and the conditions and then infer any relationship between any operator and any condition. The descriptions of the concepts need to be semantically rich enough to make possible the automatic selection of operators and algorithms using inference.

One challenge is to harmonise the differing definitions of operators described earlier (section 2.7). This will be done by starting with existing descriptions of operators in the literature. Take for example, the definition of an *Aggregate* operator by Roth et al. (2011):

“The aggregate operator captures the spatial extent of multiple features with a single feature of increased dimensionality (i.e., lines-to-polygon, points-to-polygon, or points-to-line)” (p43).

They also define a *merge* operator:

“The merge operator combines an array of related features into a single representative feature without a change in dimension” (p44).

From these two definitions we can determine that the ontology requires the concepts of an *operator*, *feature* and *dimensionality*. We can also conclude that some operators will *reduce* the number of features with or without a change dimensionality. We can therefore define a relationship between an operator and a feature. The two definitions cited above, however, make no mention of *why* the operator might be applied. This is necessary if we wish to automate their selection.

The McMaster and Shea (1992), definition of *aggregation* applies only to point features and takes a group of such features in close proximity and represents them with a single area feature. They also provide a reason for using the operator: the density of point features in a particular region on the map prevents them from being displayed individually. So we can state

in the ontology that aggregation⁵ reduces the density of features. If we also state that congestion can be resolved by a reduction in feature density, we can infer that aggregation resolves congestion (Figure 5.6). All that is required is a measure of high feature density to determine its presence.

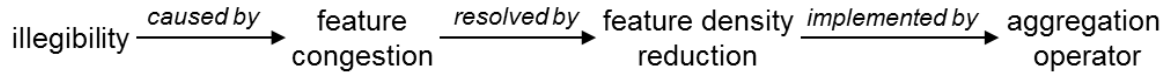


Figure 5.6 Reducing illegibility

If we consider the congested road network (Figure 1.1) then the *eliminate* operator as defined by Roth et al. (2011) – the removal of features – seems appropriate. They define three reasons why elimination may be appropriate: when there is unnecessary detail; when there are too many features, causing illegibility; and when only the most significant features are required. All three of these could be applicable to the road network. In this case we can say that the *eliminate* operator reduces the density of features. We have already asserted that a reduction in feature density resolves congestion so we can infer that the *eliminate* operator resolves congestion. This seemingly trivial example demonstrates how we can use inference to get “new expressions from old” (Davis et al., 1993, p18).

A similar process will be carried out for the commonly defined operators in the literature, encapsulating in the ontology the effects of the operator and the circumstances for its use. The general rule will be to *assert* as little as possible and *infer* as much as possible.

An operator can be implemented by one or more algorithms. The abstract concept of the operator could be omitted and the ontology asked directly to suggest an algorithm that resolves a particular geometric condition. The aim of the ontology is to automate the process of generalisation and thus the emphasis is on *machine understanding*. However, a knowledge representation, such as an ontology, is also a medium for human communication and therefore requires lucidness and neatness (Kavouras & Kokla, 2008). So, despite differences between the definitions of operators, the concept of an operator is well-understood and is used extensively in the domain. Furthermore, most algorithms are described in the literature by the operators they implement (Li, 2006). Since it is important to reflect the experts’ view of the domain in the ontology (Noy & McGuinness, 2001) the concept of an operator will be included.

⁵ This operator was termed *amalgamation* in the final version of the ontology.

5.3.2 Refining the competency questions

The competency questions can be refined in terms of the newly defined concepts. For example:

- Which measures should be applied to which mapped feature collections?
- Which generalisation operator should be applied for a given geometric condition?
- Which generalisation algorithm(s) should be applied to implement the selected operator?

The next stage is to refine and organise these concepts in a semi-formal representation using directed graphs.

5.4 Semi-formal conceptualisation

Once the important concepts in the domain have been listed and defined in natural language (Table 5.2) the next stage is to start to refine and organise these concepts. For example, the concept of an operator has been defined but it is necessary to model the different types of operators. This was done by organising the concepts into classes and defining the relationships between the classes. This can be done using the descriptions of concepts (Table 5.2). Firstly the “*is a*”, or “*type of*” or “*kind of*” subclass relationships are defined in multiple hierarchies then other relationships between classes are defined. The relationships are modelled as directed graphs where the classes are represented as nodes and the relationships as edges (Figure 5.8). Subclasses inherit the properties of their parent class. The directed graphs will be colour coded using the convention shown in Figure 5.7.

This section will focus on defining the generic concepts in then section 5.5 will focus on the specific challenges for the road accident use case. The first stage is to map out the high level concepts and relationships that are required to automatically select a generalisation algorithm (Figure 5.8). Figure 5.9 illustrates how the model may be implemented using lower level concepts (which will be described later).

The concept of a *Remedy* is introduced to act between an operator and a symptom. This helps to explain why a particular operator can resolve a particular condition. The *condition* concept is represented implicitly as part of the *ProblemFeatureCollection* class. Symptoms are measured by a *MeasureAlgorithm*. If the symptom exists then a *Remedy* that resolves it can be identified, for example a *FeatureCountReduction* will remedy *HighFeatureDensity* (Figure 5.9). From then an appropriate operator and then an appropriate algorithm can be determined.

For example, *Amalgamation* or *SelectionByAttribute* can resolve *HighFeatureDensity* in point data (Figure 5.9).

The next stage is to define the top level concepts in more detail and describe the subclasses illustrated in Figure 5.9 such as *Amalgamation* and *SelectionByAttribute*. The ontology broadly has two main parts, one that will describe the features that need to be mapped, which will be similar to a conventional domain ontology, and the other describing the more abstract concepts, such as conditions, operators, and algorithms, described in the model of McMaster and Shea (1992). The hierarchy of geographic feature types to be mapped will be considered first.

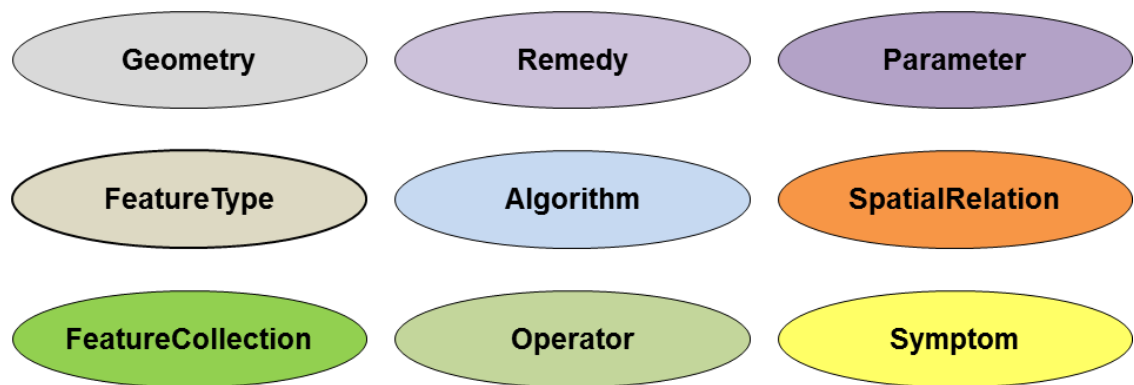


Figure 5.7 Colour coding for high level concepts

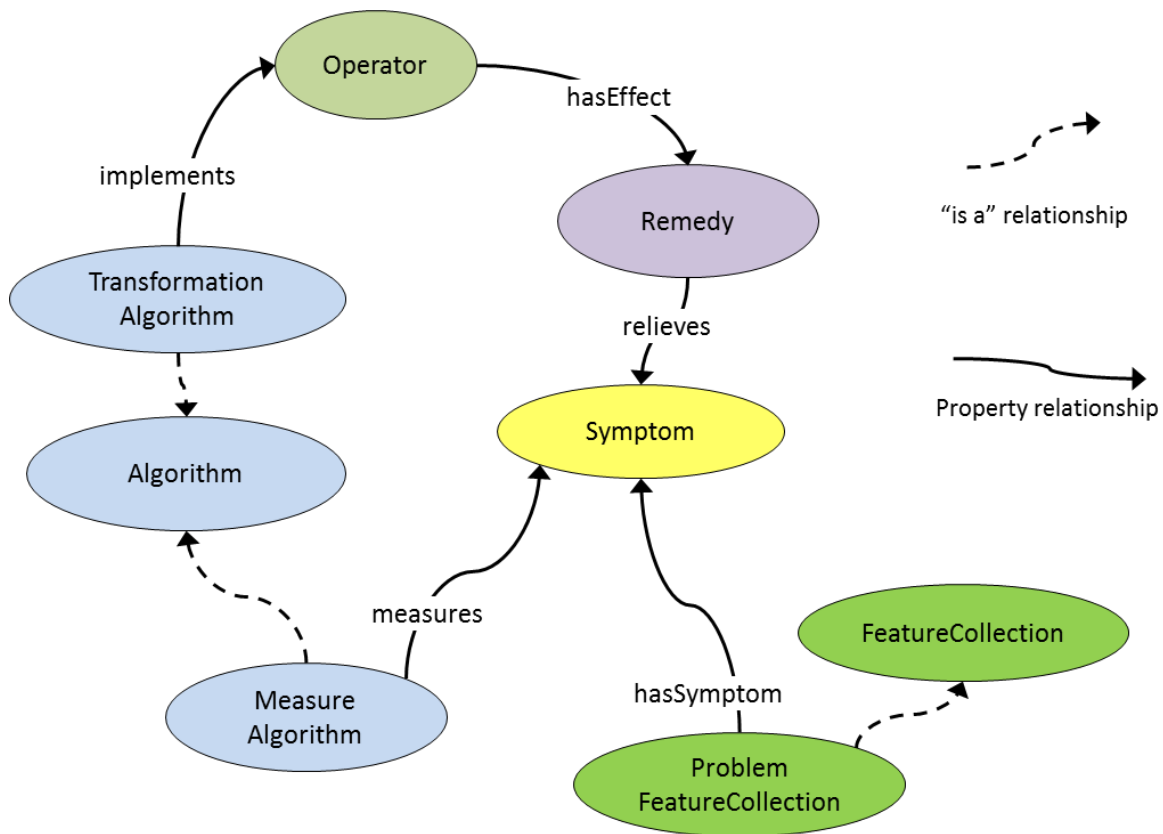


Figure 5.8 Selecting an algorithm – general case

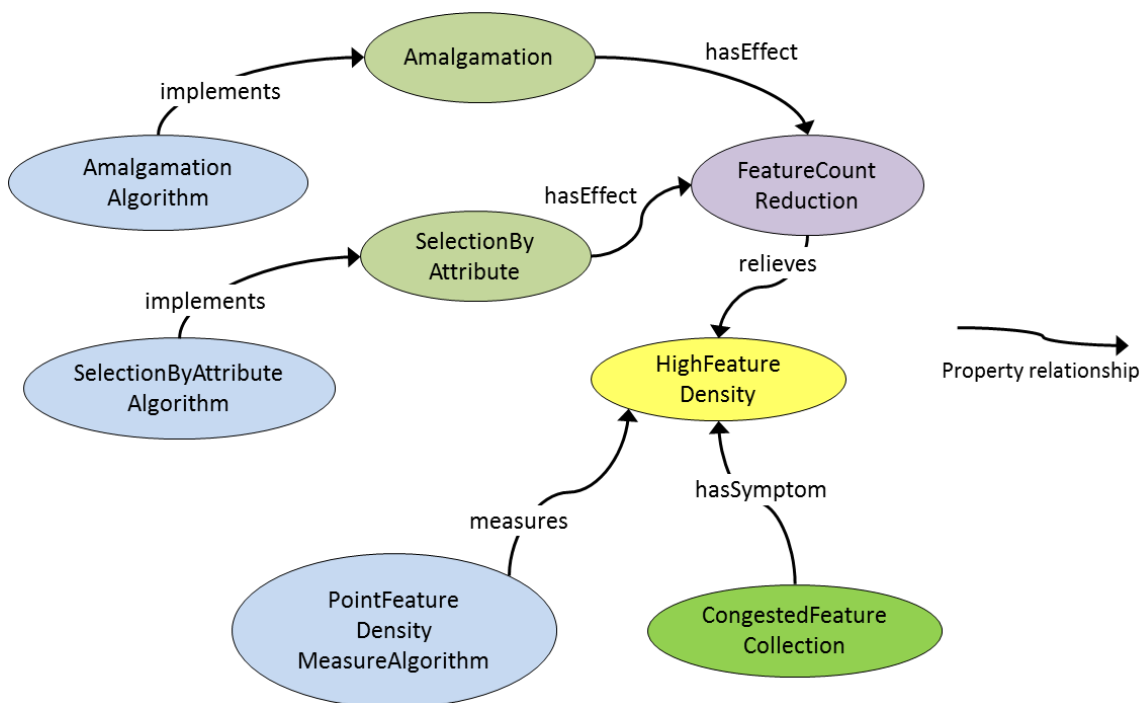


Figure 5.9 Selecting an algorithm – particular case

5.4.1 Modelling feature types in the ontology

The representation of features types in the ontology is similar to a conventional domain ontology. There are a number of approaches to modelling hierarchies of concepts in a domain ontology (Noy & McGuinness, 2001). The *top-down* approach starts with general (top-level) concepts and then considers specialisation. The *bottom-up* approach starts with the bottom-level concepts and groups them into general concepts. The method used here is the *combination* or *middle-out* approach (Sure et al., 2009), which starts with a top level concept and a number of bottom-level concepts and links them via mid-level concepts.

For example, we have a top-level *feature type* concept and a bottom-level *accident* feature type. The two concepts can be linked via a *thematic* feature type, for *immaterial* entities, and an *event* sub-type (Figure 5.10). A mid-level concept *topographic* feature type includes *material* entities such as roads and rivers. Subclasses will inherit the properties of parents.

The distinction between thematic and topographic features facilitates the task of selecting an appropriate operator. For example, if two *topographic* features overlapped, due to a change in scale but not symbology, then some action would be required such as displacing one or both of the features. However, the user will accept some degree of overlapping for *event* features such as accidents.

A distinction also needs to be made between different types of topographic feature since dedicated algorithms have been developed for generalising specific feature types such as rivers, roads, and buildings (Stanislowski & Savino, 2011; Chen et al., 2009; Yan et al., 2008). So it is not sufficient to simply search for an algorithm that generalises a network of linear features, say. The first iteration of the feature type hierarchy is depicted in Figure 5.10 (the *building* and *crime* feature types are not necessary for the use case, they are included for illustration).

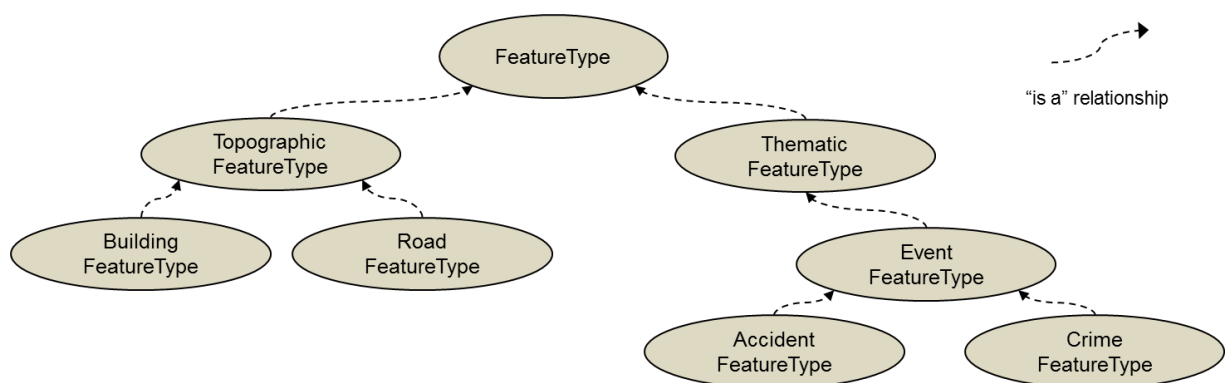


Figure 5.10 The feature type class hierarchy

In an ontology, everything must add value (Hart & Dolbear, 2013) and the addition of the *TransportFeatureType* class to the hierarchy (Figure 5.11) may seem unnecessary if it does not help answer any of the competency questions. However, as has been stated, one of the roles of any knowledge representation is to aid human communication (Kavouras & Kokla, 2008) so its inclusion is justifiable if it makes the ontology more readable by balancing the width and the depth of the hierarchy.

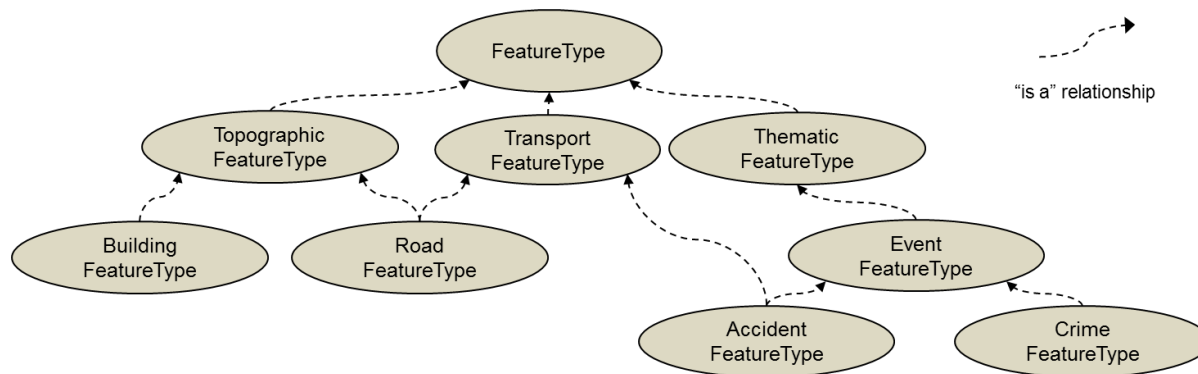


Figure 5.11 Extended feature type class hierarchy

An ontology, in contrast to an object-oriented structure, can exhibit multiple inheritance. For example the *RoadFeatureType* is both a subclass of the *TopographicFeatureType* and the *TransportFeatureType* (Figure 5.11).

So far the discussion has been limited to class and subclass or *subsumption* relationships. The advantage of ontologies is that they can represent other types of relationships. For example, the relations between feature collections (Figure 4.13) are *partonomic* relationships where a *problem* feature collection (FC) is part of a *mapped* feature collection, which is part of a *source* feature collection. Such relations are *transitive*; if a problem FC is part of a mapped FC and the mapped FC is part of the source FC then we can say that the problem FC is part of the source FC.

In fact, an ontology can describe any relation. For example, since each feature collection contains features of the same feature type, we can define a *hasFeatureType* relation between a feature collection and a feature type. Such relations are termed *property relations* and are displayed as labelled edges (Figure 5.12). It is this ability to represent relations other than hierarchical that allow connections to be made between disparate concepts such as conditions, symptoms, and operators.

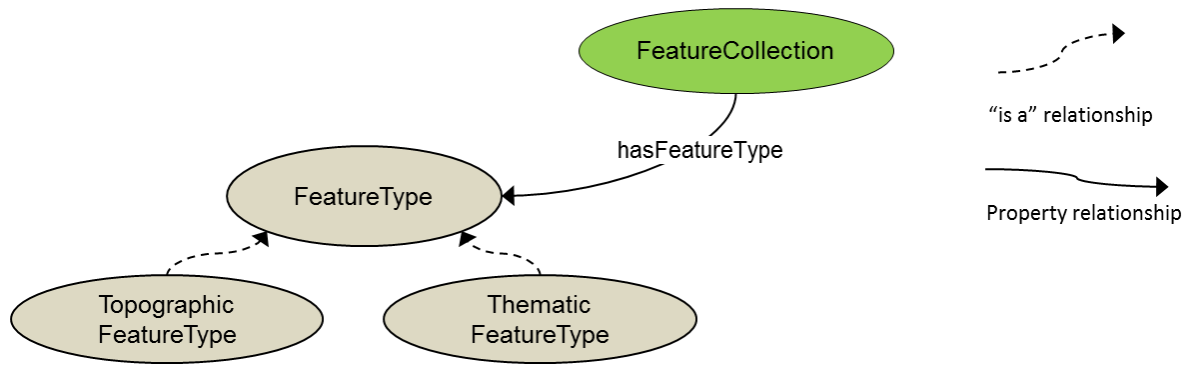


Figure 5.12 Property relations between concepts

5.4.2 Modelling operators in the ontology

The informal conceptualisation identified the *operator* as a high-level concept in the ontology and the next step is to define the lower level concepts (a *top-down* approach). As previously discussed (section 2.7) there have been a number of attempts to classify and define operators; Roth et al. (2011) identified 13 taxonomies prior to their own, with the earliest from 1963. The taxonomies often vary in the terms they use to describe the same operators (synonymy) and often use the same term to describe different operators (polysemy). There is also some disagreement over which functions can be classed as generalisation operators (Roth et al., 2011; Foerster et al., 2007a; McMaster & Shea, 1992). Given the different descriptions of operators it is necessary to choose which definitions to include in the ontology and decide how they should be organised.

The taxonomies of operators examined earlier group operators into different subclasses. For example, Roth et al. (2011) group operators into four subclasses; *geometry*, *content*, *symbol* and *label*. Such hierarchies, which aid human understanding but are not necessary for machine understanding, can be included or not. In this research they will be omitted and each type of operator will be a direct subclass of the *operator* class. This solves the problem of scope in that as long as the ontology describes all of the operators necessary to produce the desired output then whether an operator is classified as a generalisation operator or a pre-processing operator is unimportant.

The first step is to discover the characteristics that can be used to define and distinguish operators. The characteristics were drawn from the literature and grouped and described in Table 5.3. The *Effect* category is equivalent to the ontology *Remedy* class (Figure 5.8). The next step is to decide which of these characteristics each operator, in each classification, has. Figure 5.13 shows a selection of operator characteristics from the three taxonomies studied

earlier (section 2.7). If an operator definition has a particular characteristic then the value of the relevant cell is set to 1.

Each operator definition will have a *characteristic signature* which can be used to compare definitions. For example, if two operators from *different* classifications have the same signature then we can conclude they represent the same concept. Conversely, no two operators in the *same* classification should have the same signature and if they do then it implies that there needs to be a refinement of the list of characteristics, assuming that the definitions have not been misunderstood.

An easy way to compare characteristic signatures is to convert the binary number derived from the signature into a decimal value, the *total points* column in Figure 5.13. For example, it can be seen that the definition for *merge* in the Roth et al. (2011) taxonomy matches the *amalgamation* definition in the Foerster et al. (2007a) taxonomy. Therefore these two synonyms require only a single entry in the ontology. It might be beneficial to record the fact that a concept has multiple labels and that can be accommodated by the ontology (Stevens & Lord, 2012).

The process of defining the list of characteristics is iterative. For example, the *smoothing* and *simplification* definitions of the McMaster and Shea (1992) taxonomy currently have the same signature (Figure 5.13), which means that the set of characteristics needs further refinement (Table 5.3). Also each operator should have at least one *effect* characteristic; otherwise there would no reason to use it. For example, neither the *exaggeration* nor the *enhancement* operator in the McMaster and Shea list have any effects and have the same signature. In this case at least one new effect characteristic needs to be defined, so that the model distinguishes the subtle differences between these two operators. After any new additions all of the other operator definitions need to be reassessed to see if they possess the new characteristic. *Enhancement*, as defined by Foerster et al. (2007a), has a wide definition and includes *exaggeration*, *smoothing*, *squaring*, and *emphasis*. They maintained that there was not a sufficiently sophisticated cartographic model that allowed the components of the operator to be separated out. The technique of defining characteristic signatures for the operators will allow this.

Category	Characteristic	Description
Input Geometry	Point Line Polygon	The input geometry on which the operator works. At least one of these must be selected.
Level applies to	Single feature Multiple features	This is the scope at which the operator works (Li, 2006). In some cases its application is obvious. For example, it is not possible to <i>amalgamate</i> a single building. Other applications are less obvious. For example, it is possible to displace a single feature but it is done with respect to other features so the displacement operator is defined as operating on multiple features.
Effect	Reduction in detail	Indicates whether the application of the operator results in a reduction in the detail of a feature.
	Abstraction	Indicates whether the application of the operator results in an abstraction. All generalisation is an abstraction (Li, 2006) but this characteristic relates to a semantic abstraction i.e. a change of meaning. For example, a set of buildings that are amalgamated are no longer a set of buildings but a new feature representing a city block or built-up area.
	Feature count reduction	Indicates whether the application of the operator results in fewer features.
	Reduction in dimensionality	Indicates whether the application of the operator results in a decrease in dimensionality. For example, when the <i>collapse</i> operator transforms a polygon to a line.
	Increase in dimensionality	Indicates whether the application of the operator results in an increase in dimensionality. For example, the <i>combine</i> operator as defined by Foerster et al. (2007) transforms point features into a polygon.
	Change in location	Indicates whether the application of the operator results in a change in location of a feature.

Table 5.3 Operator characteristics

		Input Geometry			Scope		Effect						total points
Term		point input	line input	polygon input	single feature	multiple features	reduction in detail	abstraction	feature count reduction	reduction in dimensionality	increase in dimensionality	change in location	
McMaster & Shea	Simplification	0	1	1	1	0	1	0	0	0	0	0	928
	Smoothing	0	1	1	1	0	1	0	0	0	0	0	928
	Aggregation	1	0	0	0	1	1	1	1	0	1	0	1146
	Amalgamation	0	0	1	0	1	1	1	1	0	0	0	376
	Merging	0	1	0	0	1	1	1	1	0	0	0	632
	Collapse	0	1	1	1	0	1	0	0	1	0	0	932
	Refinement	1	1	1	0	1	0	1	1	0	0	0	1880
	Exaggeration	1	1	1	1	0	0	0	0	0	0	0	1920
	Enhancement	1	1	1	1	0	0	0	0	0	0	0	1920
	Displacement	1	1	1	0	1	0	0	0	0	0	1	1857
	Classification	1	1	1	1	0	0	1	0	0	0	0	1936
Roth et al.	Merge	1	1	1	0	1	1	1	1	0	0	0	1912
Foerster et al.	Amalgamation	1	1	1	0	1	1	1	1	0	0	0	1912

Figure 5.13 Selected characteristics from operator taxonomies

One characteristic not listed is the *output* geometry of the operator. Unlike *input* geometry, the output geometry of an operator is not relevant to the selection of that operator. However, the output geometry of one operator will affect the selection of subsequent operators. For example, if the effect of a particular operator was to produce point features then the *collapse* operator would not be subsequently applicable. In any case, the output geometry can be inferred from the input geometry and the increase/decrease in dimensionality characteristics.

The process of characterising the operators is not always straightforward; some are easier to characterise than others and the definitions of the *effects* have to be clear. For example, *refinement* as defined McMaster and Shea (1992) calls for the depiction of a “selective number and pattern” (p59) of features where they are too dense to show all of them. This is achieved by removing the smallest or least important features. Does this involve a reduction in detail? When we consider the overall map then there is certainly a reduction in detail, but when we consider individual features there is not; those features that have been retained are unchanged. So the definition of reduction in detail refers only to individual features.

The column of operator names (Figure 5.13) is headed *term* rather *operator* to emphasise that an operator is defined by its characteristics and not its label. Ultimately the labels assigned to

different operators will be invisible to the end-user and irrelevant to the machine. All that is necessary is that there is at least one operator that will remedy any given condition.

The characteristics listed in Table 5.3 and applied in Figure 5.13 proved sufficient for the use case but the list will require further refinement. For example, the description of the input data includes point, line and polygon geometries but does not include concepts such as a *network* or a *surface*. Also the technique can be used to determine if two operator definitions match *exactly* but there is currently no easy way of determining which definitions are *similar*, without examining individual characteristics. A method for highlighting similar operator definitions would be a useful aid to determining definitive representations.

Once the characteristics of the operators have been defined they can be modelled, as before, by a directed graph. Figure 5.14 depicts, as an example, the conceptualisation of the *collapse* operator, which has a definition that is shared across the three taxonomies studied.

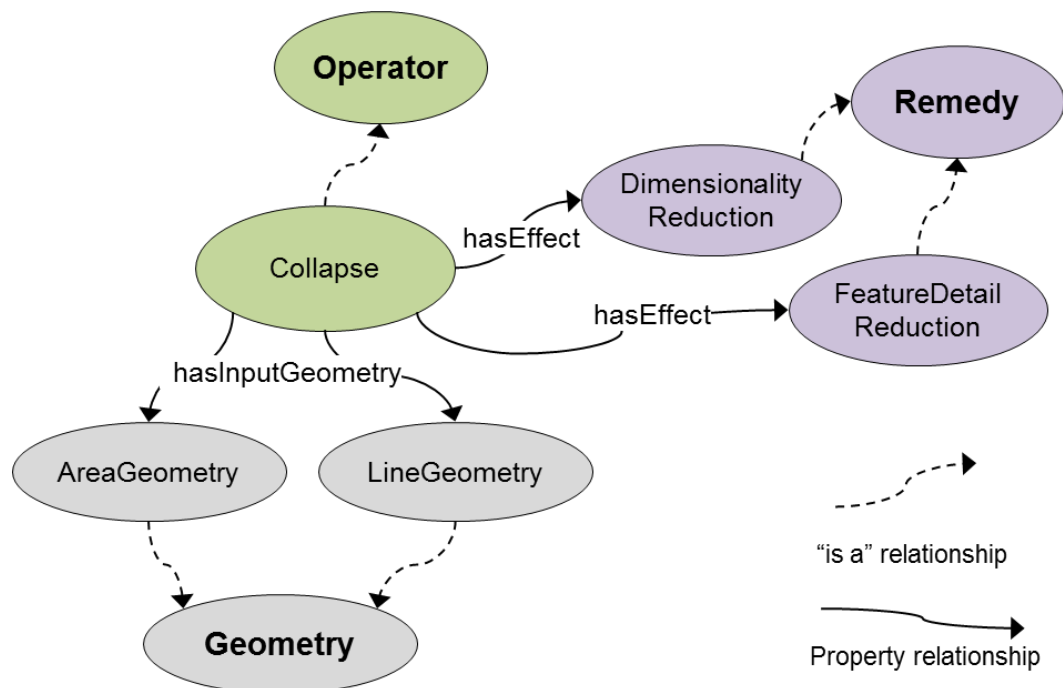


Figure 5.14 Semi-formal conceptualisation of the *collapse* operator

One of the major problems at this stage was that it was not always apparent whether a concept should be modelled as a class or an individual⁶. There are guidelines for helping the decision making (Noy & McGuinness, 2001; Stevens & Sattler, 2013) and context is an important factor. For example, consider the book “Great Expectations”. In the context of a reading list for a university course then it is an individual in the class *book*. However, in the context of a bookshop it is a class and the individuals are the copies in stock.

⁶ The term *individual* is used in OWL and is interchangeable with the term *instance*.

Geometry has been identified as a key concept and a characteristic of the operator class. Features are defined by their geometry and the application of operators is determined partly by geometry; for example, it is possible to *simplify* an area or line feature, but not a point feature. So modelling geometry correctly is important. There are two options. Geometry could be defined as a single class with three individuals (instances) – *point*, *line* and *area*. Alternatively *point*, *line* and *polygon* could exist as subclasses of the geometry class and the individuals would be geometries of particular features or feature collections. Noy and McGuinness (2001) suggest that subclasses should be introduced if they have additional properties from the superclass or if they take part in different relationships from the superclass. They also suggest that if the concept is important, then it should be introduced as a subclass. For this reason the three types of geometry were modelled as subclasses.

Generalisation is a transformative process and it is important that those changes are transmitted as the workflow progresses. For example, if we define collapse as change in geometry from polygon to point or polygon to line, then that change to the feature collection needs to be recorded in the ontology since it will affect what operators are applicable for subsequent transformations.

5.4.3 Modelling transformation algorithms

The ultimate aim of the ontology is the implementation of the McMaster and Shea (1992) generalisation model and the automatic selection of the appropriate generalisation algorithms to remedy geometric conditions. A survey of the literature to identify the characteristics of generalisation algorithms, similar to that of the operator taxonomies, was carried out. The process was, by necessity, iterative and the characteristics identified in Table 5.4 were developed from a starting list that was refined as more algorithms were examined. The survey focussed on algorithms for point aggregation, line simplification and line smoothing, road selection and building amalgamation. This ensured that algorithms for the generalisation of points, lines and polygons were surveyed. Point aggregation and road selection algorithms were surveyed since they were required for the road accident mapping use case. Not all of the characteristics listed in Table 5.4 were implemented in the ontology and some are included to aid human comprehension; specifically the “Reference” and “Li Category” characteristics were not modelled.

Most algorithms have one or more parameters. Once the algorithm has been selected then any parameter values will have to be calculated automatically for on-demand mapping. The survey was therefore extended to establish the characteristics of parameters (Table 5.5). Over

forty journal and conference papers were examined in the survey and Table 5.6 shows an example.

Characteristic	Description
Name	Common name of the algorithm e.g. <i>Douglas-Peucker</i> .
Reference	Journal/conference paper where the algorithm is first described.
Operator(s)	The operator that the algorithm implements. The operator name recorded is that used in the reference. Some algorithms implement more than one operator; line simplification <i>and</i> smoothing, for example (McMaster, 1989).
“Li” Category	<p>Category of the algorithm based on Li (2006) (see Figure 2.7). For example, <i>aggregation of a set of point features</i>. This categorisation was used in addition to <i>operator</i> because of the differences in the names of operators that are used to describe the function of algorithms. For example, the process of removing minor roads from a network has been termed <i>Simplification</i> (Regnault & McMaster, 2007), <i>Thinning</i> (ESRI, 2012) and <i>Selection</i> (Touya, 2010).</p> <p>Since the “Li” categories are relatively expressive and based on algorithms and not operators they can provide a better description than the operator category.</p>
Description	Natural language description of the algorithm.
Application	The context where the algorithm has been used. Some algorithms have been developed for specific contexts, such as generalising rural buildings (Revell, 2004).
Input features	Nature and geometry of source data. For example, set of points, single line.
Output features	Nature and geometry of output data. e.g. polygon representing a cluster of points.
Target scale	Some algorithms have been developed for a particular target scale (Benz & Weibel, 2013; Regnault & Revell, 2007; Chaudhry & Mackaness, 2005).
Source scale	Some algorithms have been developed for a particular source scale although determining this is not as easy as determining the target scale since source data is frequently held in spatial databases where the “scale” of the data is not always obvious.

Table 5.4 Characteristics of generalisation algorithms

Characteristic	Description
Name	Name of the parameter e.g. <i>maximum distance</i> .
Description	Natural language description of the parameter.
Role	What the role the parameter plays. For example, the parameter might affect the amount of generalisation or the time taken to execute the algorithm.
Effect on output	Describes the particular impact of the parameter.
Necessity	Indicates whether a parameter can be omitted or whether a default could be provided.
Weight	Defines the importance of the parameter in relation to the other parameters of the algorithm. For example a parameter that effects the amount of generalisation will be given a higher weight than one that servers a stop on the number of iterations. The combined weights of an algorithm's parameters will sum to 1. This is a subjective assessment.
Data type	Double, integer etc.
Units	Linear units, areal units etc.
Range	Range of possible values for the parameter.

Table 5.5 Characteristics of algorithm parameters

It was immediately apparent from the survey that each operator is implemented by a number of algorithms. For example, six algorithms were identified for building amalgamation (ESRI, 2011a; Damen et al., 2008; Yan et al., 2008; Regnauld & Revell, 2007; Regnauld, 2003; Revell, 2004). There are a number of reasons for this variety. For example, many algorithms have been developed or refined to solve specific problems such as amalgamating rural or urban buildings or smoothing mountain roads.

The context in which the algorithm is applied is important. Some algorithms such as the line simplification algorithm of Douglas and Peucker (1973) do not specify a particular context, other are highly specific, specifying the target scale and the feature type of the source data. For example, Regnauld and Revell (2007) describe two algorithms for amalgamating buildings at 1:50000 scale for both rural and urban contexts. The building simplification algorithm described in Table 5.6 is aimed at simplifying the complex, overly-detailed building footprints that might be the result of surveying techniques such as LIDAR.

Name	Fan & Meng							
Reference	Fan and Meng (2010)							
Operator(s)	Simplification							
Li Category	Transformation of a single area feature							
Description	Building simplification – pre-processes the ground plan by removing any redundant non-characteristic points using a buffer. Then simplification by the removal of sides shorter than a specified threshold. The method depends on whether the neighbouring sides are parallel or not, and if not then angles are compared to a threshold angle.							
Application	Single buildings, complex, non-rectangular. Building plans derived from LIDAR.							
Input features	Single polygon							
Output features	Single polygon							
Target scale	Not specified							
Source scale	Not specified, but will be at a scale with high detail.							
Parameters								
Name	Description	Role	Effect on output	Necessity	Weight	Data type	Unit	Range
ϵ	½ buffer width used for pre-processing i.e. removing redundant non-characteristic points.	Distance, Buffer, Pre-processing	The larger the buffer the more points are removed.	Required	0.4	Double	m	
T_s	Minimum length that is just visible at a given scale. Any lines shorter than this are removed.	Distance threshold, Minimum length, Speed	The larger the threshold the more points are removed.	Required	0.4	Double	m	1m to 20m in their tests
W_s	Angle threshold used when the two sides adjacent to a short side are not parallel.	Threshold	Does not affect removal of shortest side – just the meeting point of the two adjacent sides after removal.	Could use default	0.2	Degrees		< 20°

Table 5.6 Example algorithm characterisation

Extracting information about particular algorithms from the literature is not always straightforward. For example, rather than describe a single algorithm that implements a single operator, the literature frequently describes specific techniques that combine existing algorithms to solve a particular problem. Revell (2004), for example, describes a technique for *identifying* clusters of rural buildings at a scale of 1:50K, *amalgamating* each cluster and *simplifying* the resulting geometries.

Despite the complexities, the survey helped to identify the main characteristics of generalisation algorithms which, in turn, helped to formalise the conceptualisation of generalisation algorithms. The representation of a generic generalisation or transformation algorithm is shown in Figure 5.15.

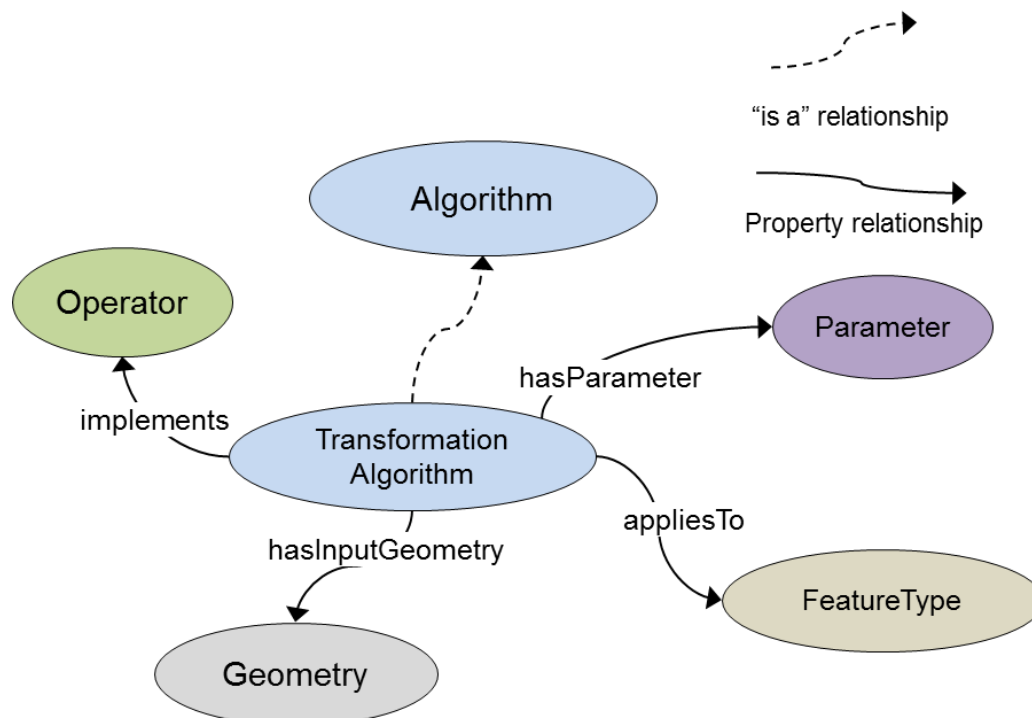


Figure 5.15 The definition of a generic transformation algorithm

A more detailed definition of two line simplification algorithms is shown in Figure 5.16. Line simplification provides a good example of why the term transformation is used since it is not universally regarded as a generalisation process. In this example, the definitions of the two algorithms will need more than a parameter to separate them. Further characteristics will need to be added to provide a reason why one should be chosen before the other.

Defining different algorithms as subclasses, such as *Douglas-Peucker* and *Visvalingam-Whyatt*, might not be easy since many algorithms are refinements of existing algorithms and there may be several versions of the Douglas-Peucker algorithm, for example. Ultimately, the automatic selection of algorithms will be based on their properties and not their origins. Therefore it might be better to represent the source of the algorithm (e.g. *Douglas and Peucker (1973)*) as a *property* of the algorithm class rather than as a subclass.

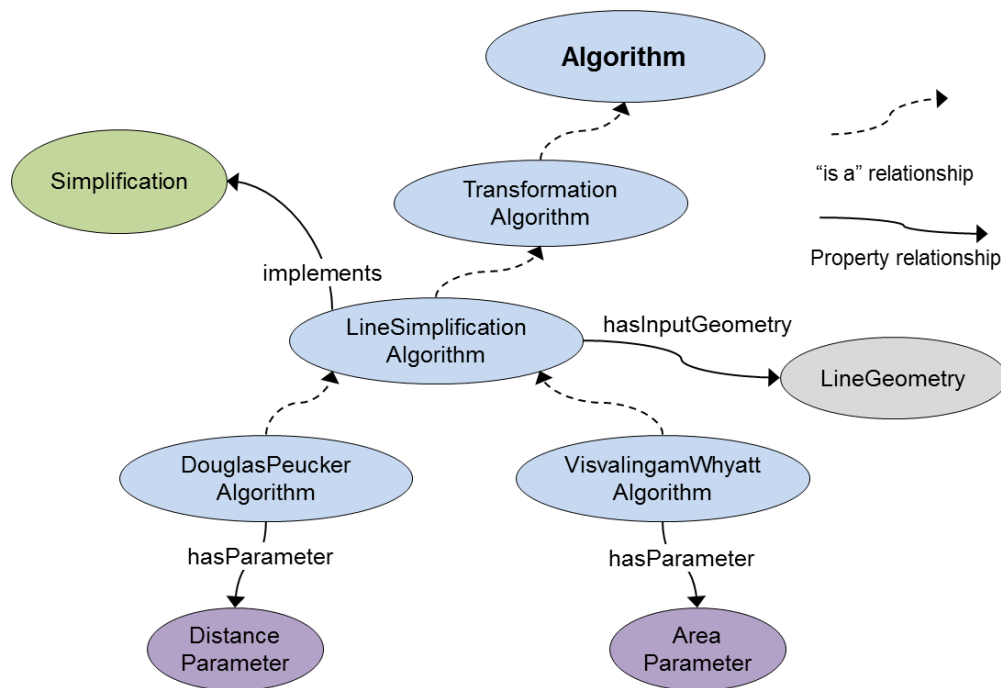


Figure 5.16 The definition of two line simplification algorithms

In this implementation the individuals (instances) of the transformation algorithm class will be *Java* functions. The Balley and Regnauld (2011a) model for on-demand mapping (Figure 4.14) specifies the exposure of algorithms via web services and this model does not prevent that.

5.4.4 Modelling measure algorithms

The role of a measure algorithm is to identify the features in the mapped feature collection (Figure 4.13) that exhibit a particular symptom (such as high feature density) of a condition (such as congestion). The definition of a measure algorithm (Figure 5.17) is less complex than that of a transformation algorithm.

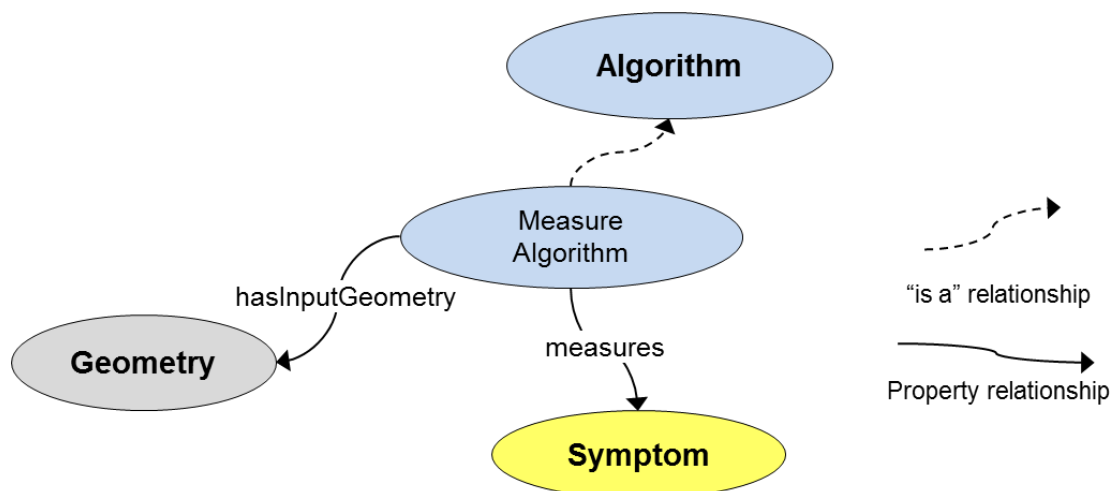


Figure 5.17 The definition of a generic measure algorithm

The concept “*generalisation*” was included in the informal conceptualisation (Table 5.2) but is omitted from the semi-formal conceptualisation since an explicit definition was not required as the concept is encapsulated by other concepts and their relationships. So far the semi-formal conceptualisation has been relatively general. The following section focuses on some of the specific challenges for the use case.

5.5 Use-case specific challenges in semi-formal conceptualisation

The previous section developed semi-formal conceptualisations of the main concepts of the ontology, in particular the *operator* and *algorithm* concepts. This section focusses on some of the specific challenges that have to be solved to implement the use case. The operators have to be defined in such a manner as to help their automatic selection. Not all operators are appropriate for the feature types in the use case. The definition of the use case highlighted a number of actions that could be used to reduce congestion in accidents and road segments (Figure 5.3 and Figure 5.4) and this section will seek to refine the operators for the use case.

5.5.1 Selection by Attribute

Perhaps the simplest way to reduce congestion of road accidents is to remove the least important accidents (Figure 5.3c). Li (2006) terms this operator *selective omission* but it is not immediately obvious what the equivalent is in the three taxonomies studied. The nearest equivalent is probably *class selection* as defined by Foerster et al. (2007a). The process is also contained within the very broad *add* operator defined by Roth et al. (2011). It may be included as part of *refinement* operator of McMaster and Shea (1992) but it is not clear, and all three definitions are too broad for what is required.

There is little detailed discussion of this process in the literature, partly because it is relatively easy to implement and partly because it is not always regarded as a generalisation process but as a pre-processing step. Unlike most generalisation operators, it has no geometric effect. It may be that since it requires the features to have attribute information it has been historically less relevant. It is only in recent years that maps have been created from databases rather than from other maps at a larger scale. The relative importance of topographic features can be inferred from their geometry, specifically their size, rather than a particular attribute value. The generalisation of topographic features can then be controlled by applying a minimum size constraint. However, the minimum size constraint is not relevant to point event features such as accidents and this is why *selection by attribute* needs to be defined as a distinct operator (Figure 5.18).

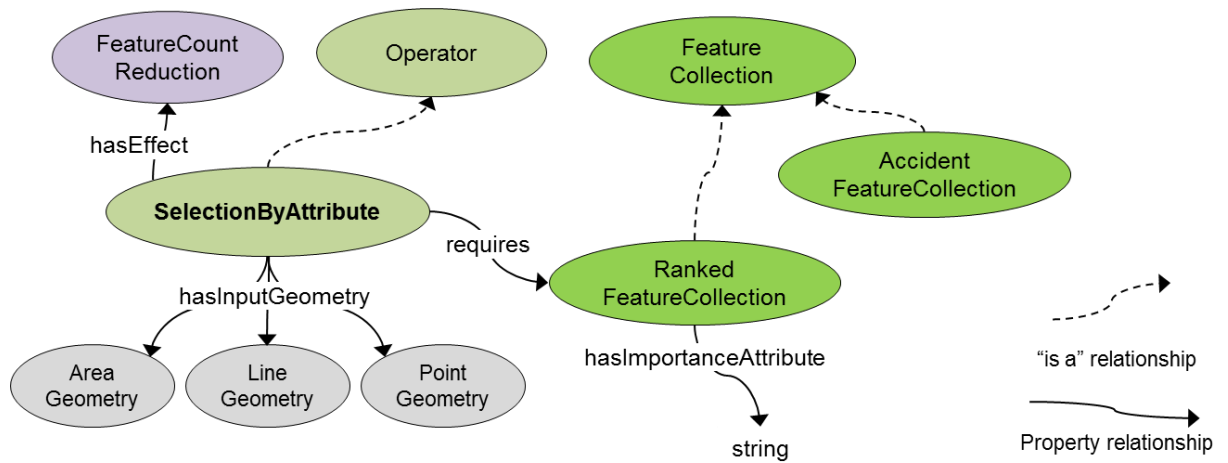


Figure 5.18 Defining the selection by attribute operator

As stated earlier (section 5.2.1), the source feature collection must have an attribute that can be used to rank the features by importance. Such a feature collection is defined in the ontology as a *RankedFeatureCollection* (Figure 5.18). The *SelectionByAttribute* operator is defined as *requiring* a *RankedFeatureCollection*. The *RankedFeatureCollection* is defined by having an importance attribute represented by a string which will simply be the name of the source dataset attribute that can be used for ranking. Such a relation is termed a *data property* in contrast to the object properties that have been defined so far. Any individual (instance) of the *FeatureCollection* class that has a value for the *hasImportanceAttribute* data property will be inferred to be a *RankedFeatureCollection*. The *hasImportanceAttribute* is not made a property of the *AccidentFeatureCollection* class since that would imply that *all* accident feature collections had an importance attribute. The dataset in the use case has an attribute *SEVERITY*, which can be used as an importance attribute, and that detail is recorded in the ontology.

The *selection by attribute* operator makes no semantic changes to the individual features; all that has changed is their number. However, if congestion in a collection of road accidents is resolved by grouping the accidents into a single feature then the features have changing meaning from a set of accident features to a hot-spot, say. The individual attributes, including the importance attribute, of the accidents are lost. This will effect subsequent transformation of the data. The ontology will therefore need to manage the change in *meaning* of a feature or feature collection. This process is termed *semantic propagation* (Janowicz et al., 2010).

One of the road segment datasets employed in the use⁷ case has an attribute that describes the *class* of the road (Motorway, A road, B road etc.) that *could* be used for ranking features.

⁷ Ordnance Survey's ITN dataset provided by the Digimap service (University of Edinburgh, 2014)

However, selection by attribute is not appropriate for the road features since it is necessary to maintain a consistent network when generalising roads and the retention of some minor roads can help to do this⁸. The removal of road segments needs to be more sophisticated (see section 5.5.4). If the dataset has a ranking attribute then the *select by attribute* operator should not be selected for use on the road network.

The solution was to define a new feature class, *NetworkFeatureCollection* and add *RoadFeatureCollection* as a subclass (Figure 5.19). If it is asserted that a network *forbids* selection by attribute then, since a road feature collection is a type of network, selection by attribute cannot be applied to roads.

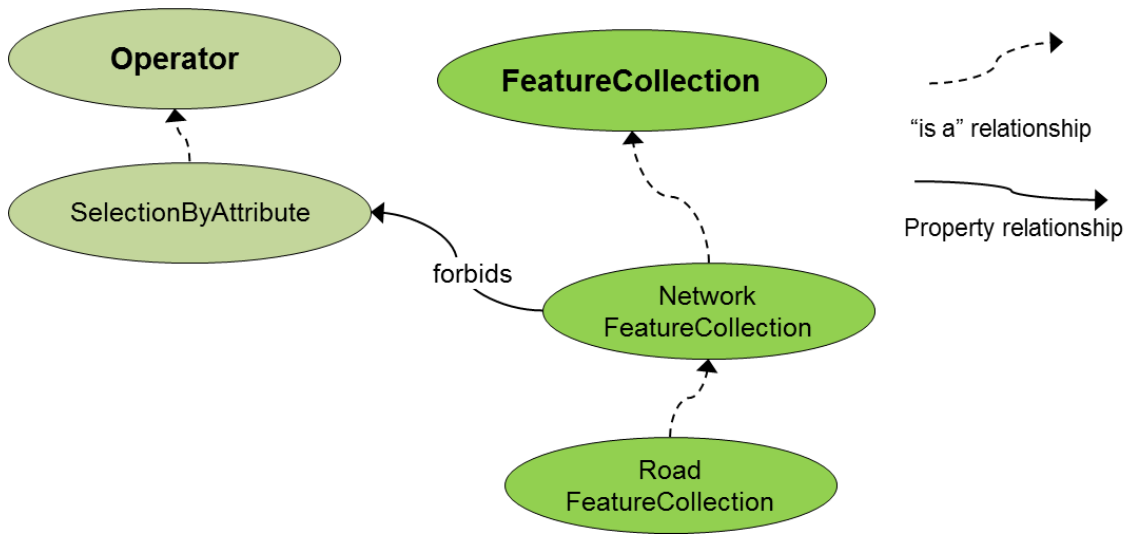


Figure 5.19 Preventing the application of *selection by attribute* to road features

5.5.2 Amalgamation

Amalgamation can be defined as the replacement a cluster of features with a single *area* feature of the same extent. Generally the literature makes a distinction between two operators depending on whether there is a change in dimensionality or not. For example, Foerster et al. (2007a) use the term *combine* when an increase in dimensionality occurs and *amalgamation* when not. Roth et al. (2011) uses the term *aggregate* when an increase in dimensionality occurs and *merge* when not. McMaster and Shea (1992) use the terms *aggregation*, *merging*, and *amalgamation* for the combination of *point* (to area), *line* (to line) and *area* (to area) features, respectively. Li (2006) terms this operator *regionalisation* when applied to point features and *aggregation* when applied to area features. The operator as defined here, termed *amalgamation*, applies to source data of any dimensionality (Figure 5.20) since there is no requirement from the use case for two separate operators. Its application to accidents is

⁸ Some road pruning algorithms will utilise the road class to aid selection but not use it as the sole criteria for selection (Zhou & Li, 2012).

relatively straightforward (Figure 5.3e) but its application to road features needs to be considered.

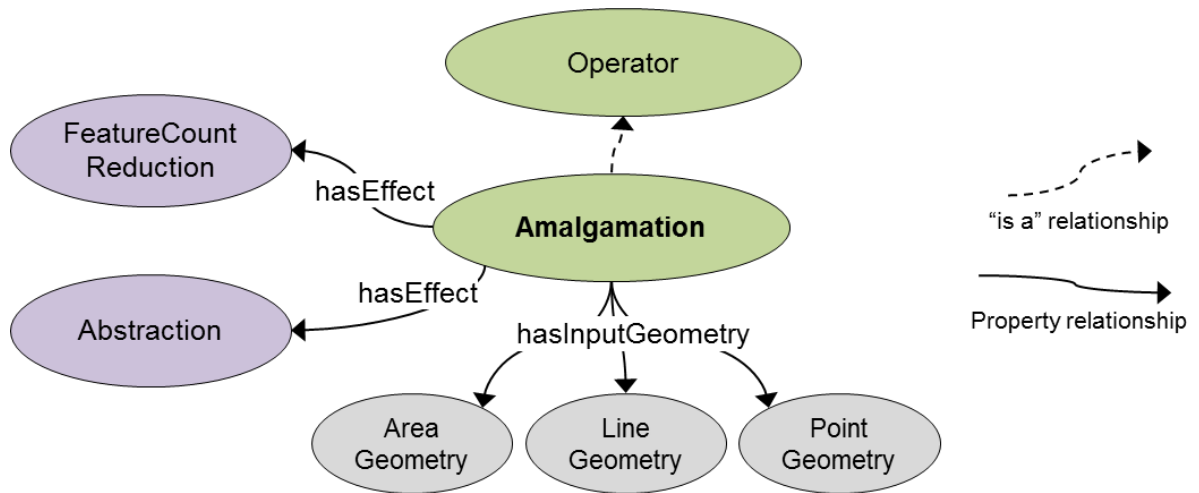


Figure 5.20 The definition of the amalgamation operator

Figure 5.21a depicts urban buildings at a scale where individual buildings are difficult to distinguish and amalgamating the buildings into “city blocks” would be appropriate. Figure 5.21b depicts road features (as polygons) that suffer the same problem. However, in this case the amalgamation of the features would be inappropriate; whereas the concept of a built-up area representing a number of individual buildings is conventional the same does not apply for roads. The ontology needs to be designed in such a way that it does not suggest this operator for road features.



(a) High density buildings



(b) High density roads

Figure 5.21 High density topographic features

The reason amalgamation is inappropriate is that it performs a semantic abstraction which is unsuitable for networks such as roads and rivers; there is no concept that describes a cluster of

roads⁹. The most direct solution would be to simply assert that the amalgamation operator does not apply to a network (as was done with the *selection by attribute* operator). However, a more generic solution is to define *Abstraction* as a new subclass of *Remedy* (Figure 5.22).

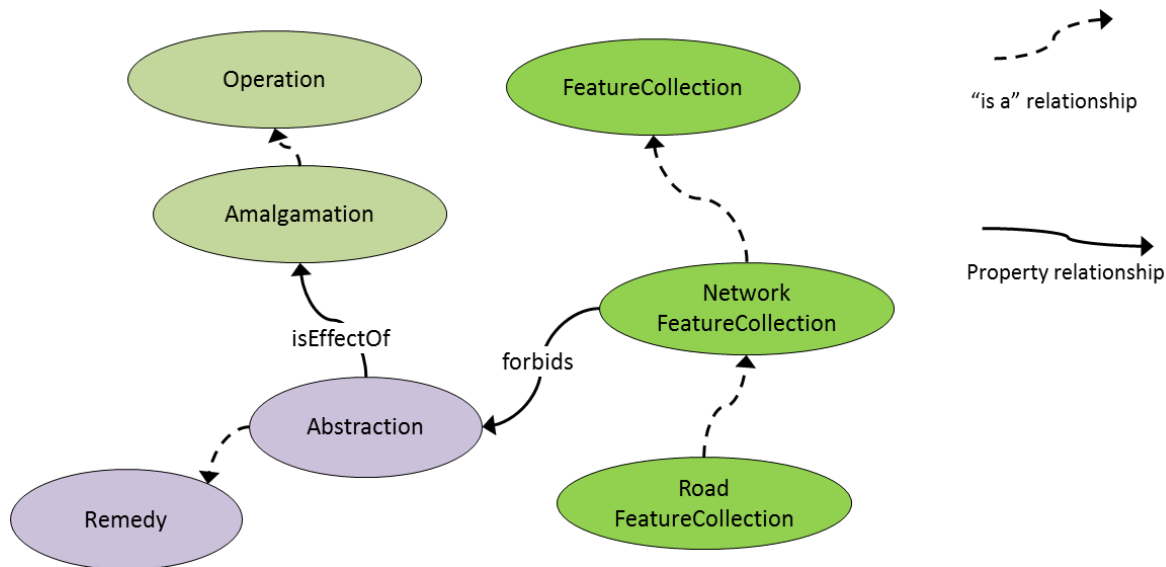


Figure 5.22 Preventing the application of *amalgamation* to road features

It is asserted that *Abstraction* is an effect of *amalgamation* and that a *Network FC* forbids *abstraction*. It is then possible to define a *property chain*¹⁰ (Figure 5.23), from which it is possible to infer that *amalgamation* should not be applied to road features.

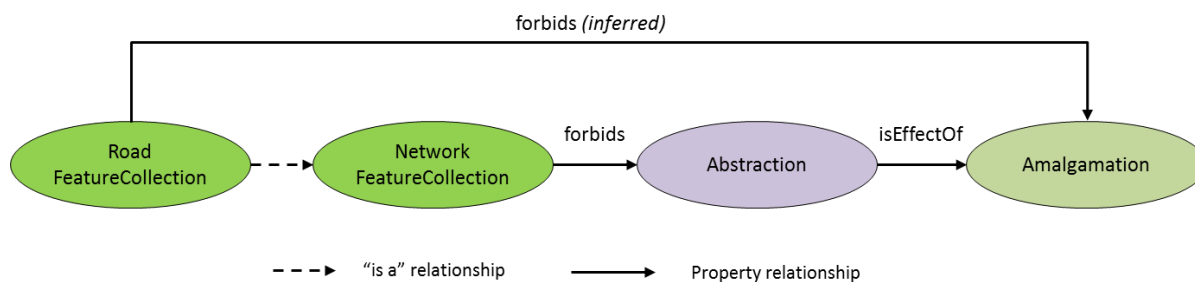


Figure 5.23 A property chain

As the intention is to use semantics, a direct assertion – that a network should not be amalgamated – was avoided. Therefore any operator that performs an abstraction will not be applied to any network of features including roads and rivers. Describing the nature of the operator, *amalgamation*, allows it to be reasoned about.

⁹ At a relatively small scale a cluster of roads could be used to define the extent of a settlement but that is normally done by buildings.

¹⁰ This is done using the property chain function of Protégé in the formal conceptualisation phase.

As discussed earlier the *selection by attribute* operator was deemed inappropriate for generalising a network but in that case a direct assertion was made (Figure 5.24). The more direct assertion is less useful since we are not getting “new expressions from old” (Davis et al., 1993, p18) but it is difficult to see how it could be refined; it is in the nature of a network that its components cannot be removed based simply on the value of an attribute.

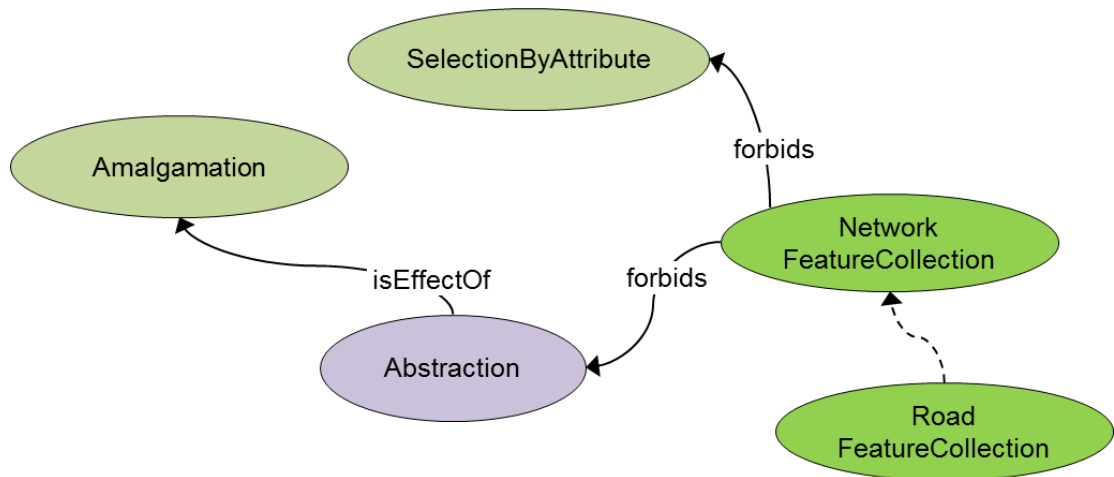


Figure 5.24 Indirect and direct restrictions on operators

5.5.3 Aggregation

The process in Figure 5.3f can be defined as “the categorisation of a set of points into groups and then representation of each group by a single point” (Li, 2006, p76) and is termed *aggregation* in the ontology. It is equivalent to the *amalgamation* operator of Foerster et al. (2007a) and the *merge* operator of Roth et al. (2011) with point data as input. This operator differs from amalgamation in that there is no change in dimensionality and it only applies to point features (Figure 5.25).

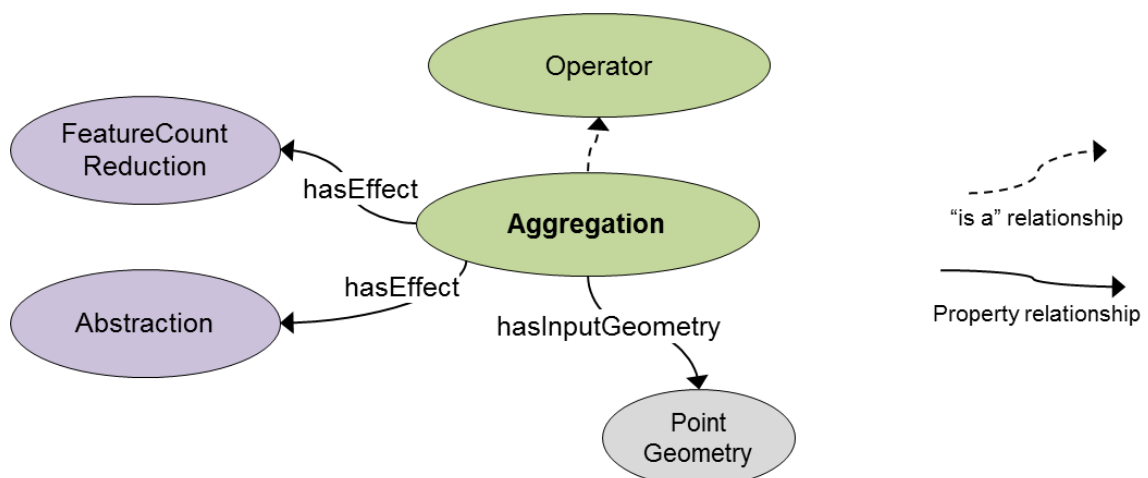


Figure 5.25 Definition of the aggregation operator

5.5.4 Pruning

The operators discussed in detail so far are applicable to the accident data. The operator discussed here, termed pruning, is relevant to the road network. It is similar to *selection by attribute* in that it selectively reduces the number of features but does it in a distinctively different manner that requires a separate operator. This operator is sometimes referred to as *refinement* (Regnauld & McMaster, 2007; McMaster & Shea, 1992) or *thinning* (ESRI, 2012) and aims to remove features from while respecting the topology of the features (Figure 5.4c).

Roth et al. (2011) define a more general operator, *eliminate*, and regard *refinement* as a type of elimination that can be managed at the algorithm level. Their reasoning is that refinement is a function of the data structure and that the effect, the elimination of features, is the same. This means that the requirement for a network is a property of the *algorithm* (Figure 5.26a) rather than the *operator* (Figure 5.26b). This raises the general question of whether the detail exists with the operator or the algorithm. It should be remembered that Roth et al. (2011) have a different motivation for their classification and here it is the operator that will have the requirement for a network.

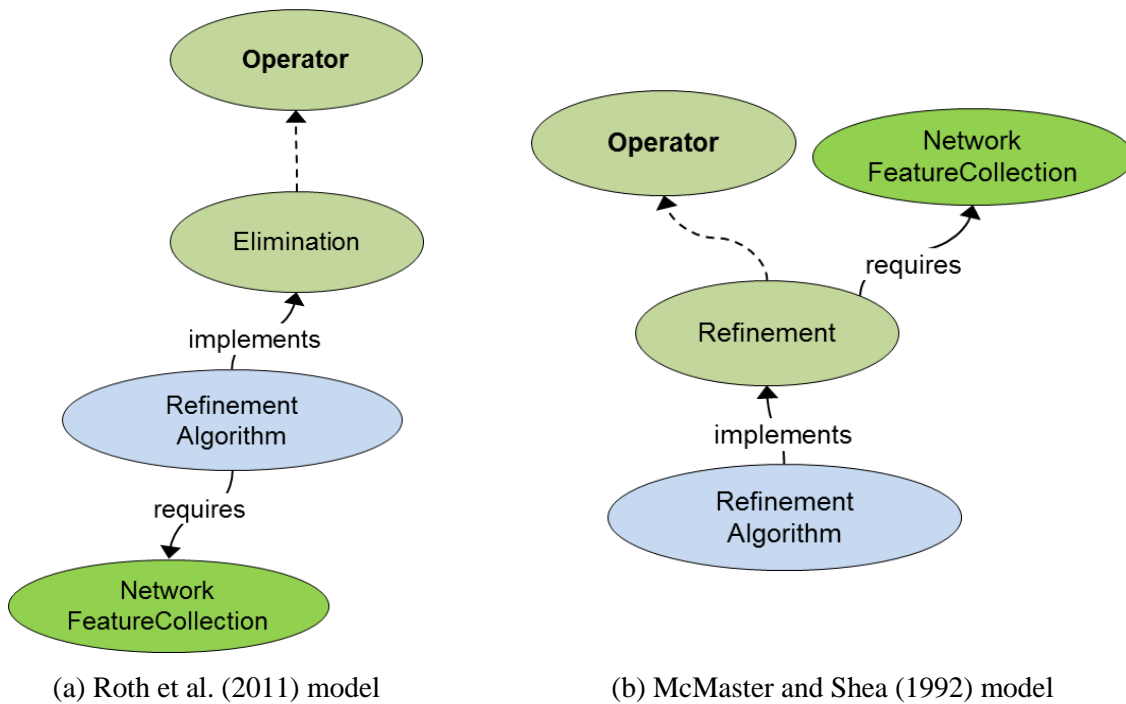


Figure 5.26 Location of properties

The term *pruning* has been used before (Stanislowski & Savino, 2011) and will be used here since it is more meaningful than “refinement”. The definition of the operator (Figure 5.27) prevents it being applied to the accident dataset since it requires a network feature collection and line geometry.

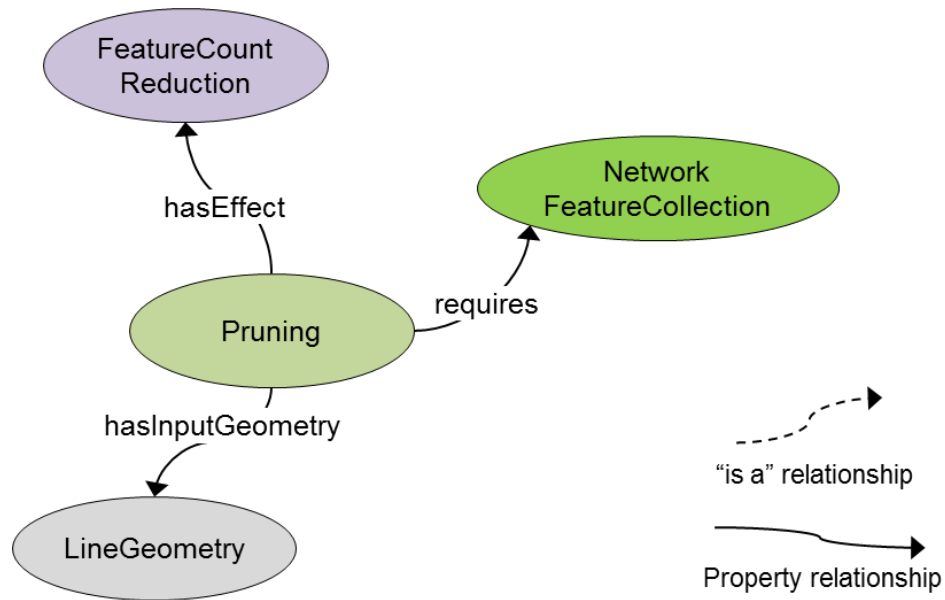


Figure 5.27 The definition of the *pruning* operator

The operators required by the use case have been semi-formally conceptualised. The evaluation phase may lead to further refinement but it is now necessary to consider the conceptualisation of the measure algorithms.

5.5.5 Selecting the correct measure algorithm

The use cases focus on feature congestion, of which *high feature density* is a symptom. Different algorithms are required to measure high feature density in accidents and in roads. The ontology will include five types of measure algorithms (Figure 5.28). Algorithms are required for the three different geometries since the accidents will be represented as points (initially) and area features (if amalgamated) and the roads will be represented as area features and line features. The accident feature density will be measured by the *generic* algorithms; *PointFeatureDensityMeasureAlgorithm* and *AreaFeatureDensityMeasureAlgorithm*. The road feature density will, however, be measured by two *specialised* algorithms; *RoadAreaFeatureDensityMeasureAlgorithm* and *RoadLineFeatureDensityMeasureAlgorithm*. The density of crossroads is used in these algorithms as a more subtle measure of the density of road features than the density of feature centroids that is used in the generic line and area feature measures (the detailed implementation of these algorithms is described in Chapter 6).

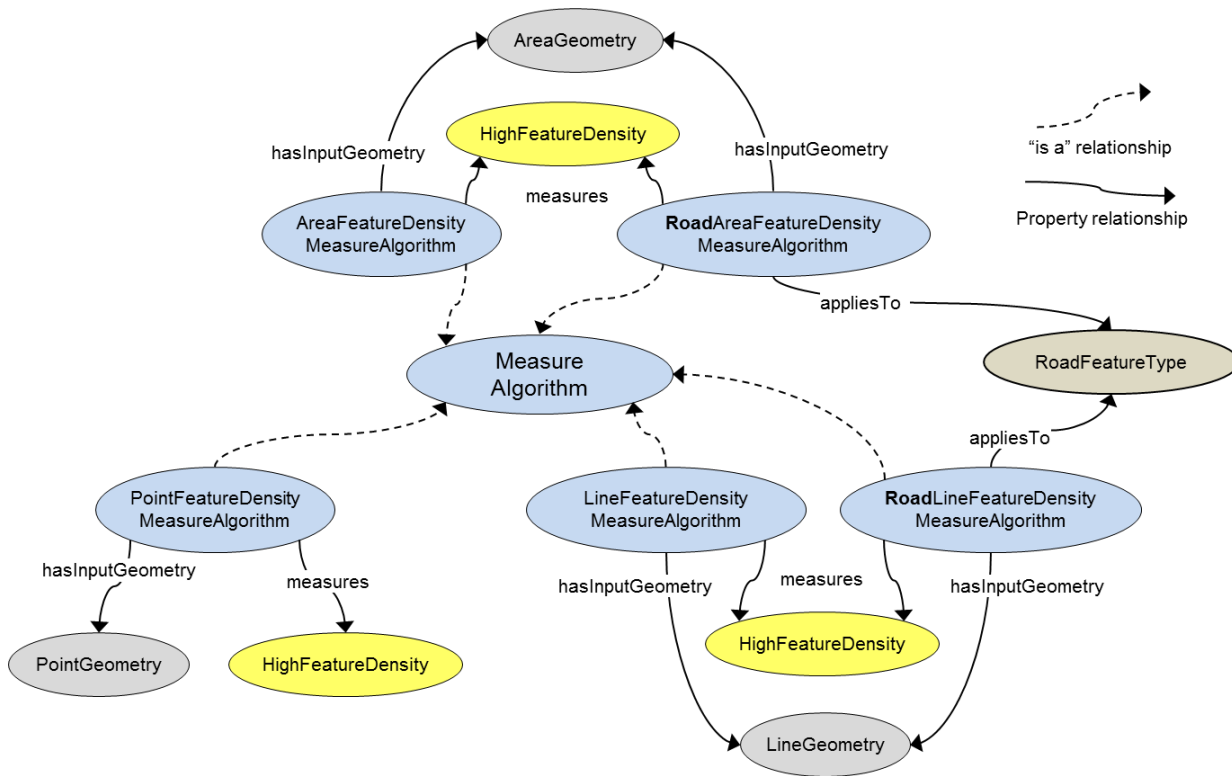


Figure 5.28 Definition of measure algorithms for the use case

The hierarchy for the measure algorithms is shallow. It could be argued that the *RoadLineFeatureDensityMeasureAlgorithm* is a specialisation of the *LineFeatureDensityMeasureAlgorithm* class and should be made a subclass of it. However, this is not the case since the *RoadLineFeatureDensityMeasureAlgorithm* only applies to road features and not any other type of line feature. The generic *LineFeatureDensityMeasureAlgorithm* is not required by the use case and no individuals (instances) will be created; that is, the algorithm will not be implemented in code. However, the class has been included as a test. The system should *not* suggest an algorithm from class for the road network.

5.5.6 Modelling spatial relations in the ontology

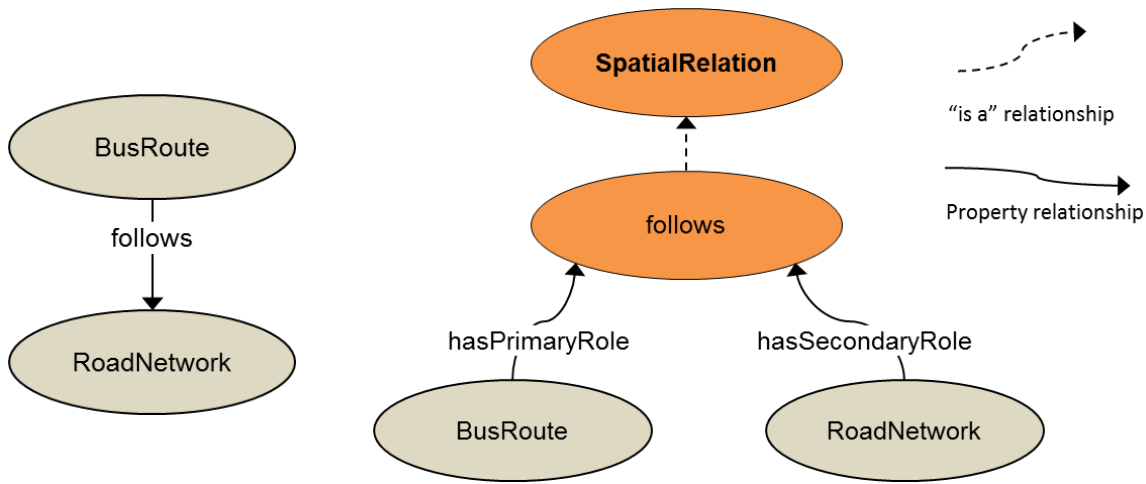
The aim when modelling spatial relations is to *assert* relations between different feature types so that they can be respected during generalisation. This in contrast to recent work by Jaara et al. (2012) where relations at the individual feature level are *inferred* and then respected during any transformation. In yet another approach, Corcoran et al. (2012) use generalisation to infer spatial relations, for example between an access road and an amalgamated group of buildings. The loss of context that occurs when generalising without respecting spatial relations is not always as obvious as in Figure 5.5. For example, the removal of the minor road segment (A)

in Figure 1.5 may lead the user to conclude that the accident hot-spot is located on a straight section of road rather than at a junction.

A spatial relation between two geographic features can either be expressed as a *metric* (a proximity relation can be measured by distance, for example) or expressed as a binary relation using *predicates*; the road is *parallel to* the rail track (Egenhofer & Franzosa, 1991).

Shi (2011) believes that the *spatial* relations between objects in ontologies have been ignored in favour of *semantic* relations but more recently there have been a number of attempts to define spatial relations in an ontology (Ordnance Survey, 2014b; Touya et al., 2014; Bucher et al., 2012; Laurini, 2012; Touya et al., 2012). The intention here is to encapsulate those spatial relations between feature types that are the result of semantic relationships. For example, a road accident is an event that takes place on a road; a bus stop is an access point to a network.

It might seem obvious that spatial relations should be modelled as properties of classes. i.e. a relation is modelled using a relation (Figure 5.29a), which is the model used by the Ordnance Survey (2014b). However, in the Web Ontology Language (OWL), properties have limited characteristics. An *inverse* property can be defined (the inverse of *follows* is *isFollowedBy*) or a *symmetric* property (*nextTo*, for example) but there is no way of adding attributes to properties. Modelling spatial relations as properties is limited to binary relations so we cannot model higher order spatial relations, such as the intersection of three different features. The solution is to model spatial relations as classes (Figure 5.29b) but at the expense of making the reasoning process more complex.



(a) Spatial relation as a property

(b) Spatial relation as a class

Figure 5.29 Representing spatial relations in the ontology

The relationship between accidents and roads can be considered initially as a *semantic* relation: by definition a road accident is an event that takes place *on* a road. However, the nature of the “on” relation is vague. Consider the different meanings between “the nose *on* your face”, “the house *on* the street”, and “the box *on* the floor” (Varanka & Caro, 2013). The relation between the accident and the road is similar to the last of these. The semantic relation is modelled simply in the ontology as a *property* (Figure 5.30). At the feature *type* level it is asserted that, in *general*, features of the type “accident” are “on” features of the type “road”. However, at the feature *collection* level, each *particular* relationship will in fact be between *individual* feature collections (Figure 5.31). Modelling this semantic relation between the two feature collections allows the system to infer that when a particular accident feature collection is to be mapped then a particular road feature collection can provide context.

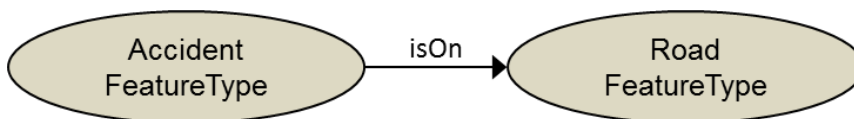


Figure 5.30 Modelling the *general* semantic relationship between accidents and roads

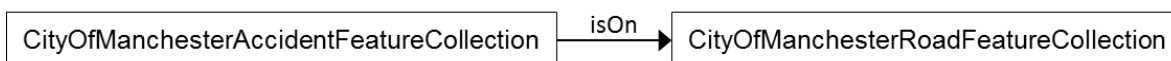


Figure 5.31 Modelling a *particular* semantic relationship between roads and accidents

To allow for the generalisation of the road network with respect to the accident data, however, the imprecise semantic relation, “is on”, needs to be expressed as a *spatial* relation. In fact, it

is expressed as a number of *spatial* relations which are dependent on the current geometries of the two feature types (Figure 5.32). For example, if the road accidents have been amalgamated into an area feature and the roads have been collapsed to lines then the relationship is *intersects* (Figure 5.32d). The relationship can be used as a constraint on the generalisation of the roads. So, if the road segment intersects an accident cluster then the road segment should be retained. These relationships need to be represented in the ontology.

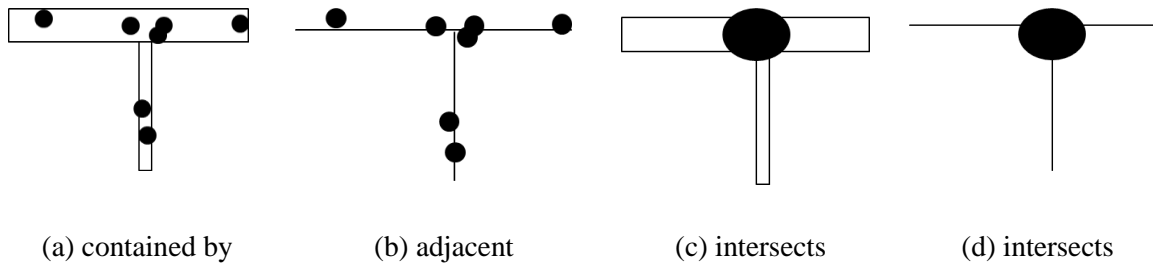


Figure 5.32 Spatial relation predicates between roads and accidents

Figure 5.33 depicts the relation *containedBy* between accidents and roads. This relation is not symmetric and the definition stresses the primacy of the accidents in the relation by referring to the accidents as *thematic* features and the roads as *support* features, using the terminology of Jaara et al. (2012). The concept of thematic and support roles is semantically more expressive than *primary* and *secondary* roles (Figure 5.29) and reflects the aim of the on-demand system to map user data (accidents) in context (road network). The *ContainedByTestAlgorithm* class will contain algorithms for testing whether a feature of one type is contained by a feature of another type. If they are (as in Figure 5.32a) then this relationship needs to be respected during generalisation. The spatial relation test algorithms are sufficiently different from the *cartometric measure* algorithms (section 5.4.4) to warrant their own class and are applicable to any feature type.

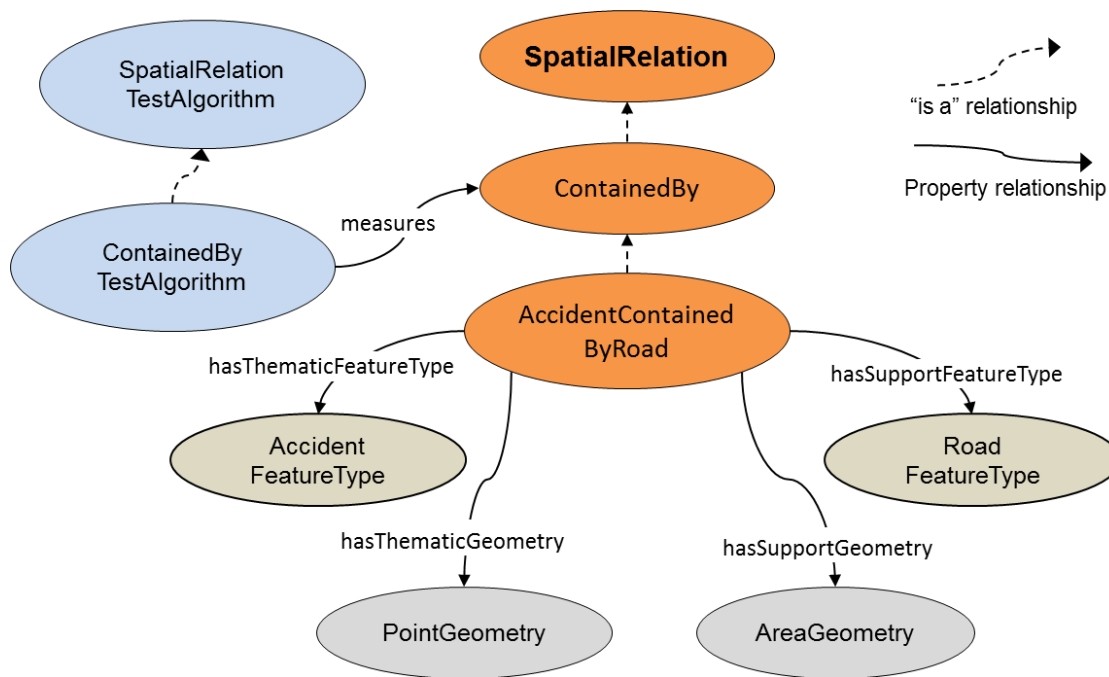


Figure 5.33 Modelling the *contained by* spatial relation as it related to accidents and roads

The relations between feature types can be used to select which features to map. For example, if the user expresses a requirement to map road accidents then the system can use the ontology to infer that the road network should also be mapped although the inverse would not be true and the *thematic* and *support* concepts help to ensure that.

Complications occur when accident features are amalgamated (Figure 5.32c and Figure 5.32d). The road segments, although generalised, retain their original character, only their geometry has changed. However, it could be argued that the accident features are replaced by a *new* feature type, an accident cluster or *hot-spot*. The original relation is between the accident feature type and roads feature type. A new feature type *accidentHotSpot* could be created and the relation between it and the road feature type defined. However, the definition of the amalgamation operator would need to be updated to record the fact that road accidents are transformed into the new *accidentHotSpot* feature type. This would be the same for any amalgamated feature type such as buildings. This change would make the ontology unnecessarily complex but the inconsistency needs to be noted. Ignoring the fact that amalgamation leads to a change in feature type, the *intersects* relationship is depicted in Figure 5.34. As with the *adjacent* relationship, and unlike the *contained by* relationship, *intersects* is symmetric. Another difference from the *contained by* relationship is that it supports two different support geometries.

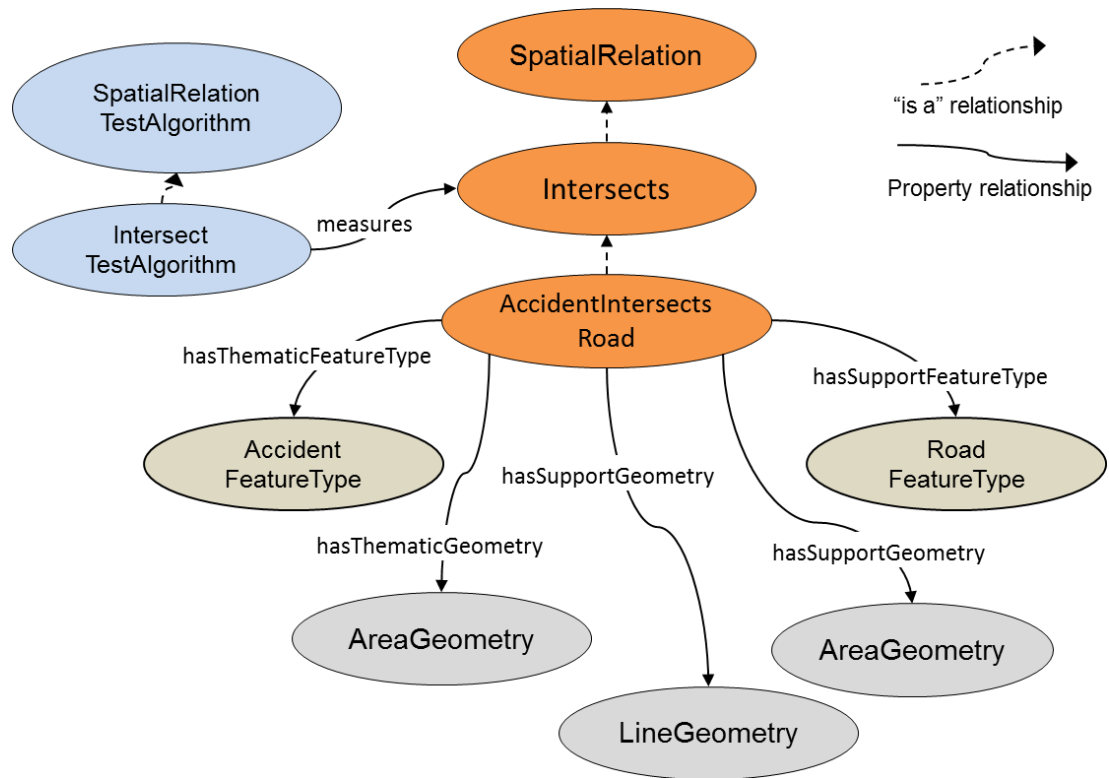


Figure 5.34 Modelling the *intersects* spatial relation in the ontology

5.6 Formal conceptualisation

The concepts defined in the directed graphs were encapsulated in a formal language (OWL-DL) using the Protégé ontology editor (Stanford Center for Biomedical Research, 2014). Protégé is a free, open-source, and widely used ontology editor. Classes and subclasses were created in Protégé, based on the directed graphs created in the previous section, and then object properties (relations) were added. Finally individuals (instances) were added.

The semi-formal stage can to some extent defer the question of whether a concept exists as a class or as an individual (instance) in the ontology. However, the question cannot be avoided in the formal conceptualisation phase. The issue, in relation to geometry was raised earlier (section 5.4.2), but it is also applicable to other concepts such as algorithms. For example, the Douglas-Peucker algorithm could be modelled as an individual in a line simplification class or as a class of its own. It is only in the evaluation stage that these uncertainties can be answered.

5.7 Evaluation

After the ontology has been formalised using OWL the competency questions can be expressed as Manchester OWL Syntax queries (Horridge & Patel-Schneider, 2008), for which Protégé has a parser. The parser requires the use of a reasoner, and HermiT (University

of Oxford, 2011), which is built into Protégé 4.1, was used. For example, the competency question

Find an algorithm that will implement a particular operator

is expressed in the Manchester OWL syntax as

```
TransformationAlgorithm and implements some Collapse and hasInputGeometry
some AreaGeometry
```

This will return any subclasses such as the *AreaFeatureCollapseAlgorithm* (Figure 5.35) and any individuals in those subclasses that meet the query.

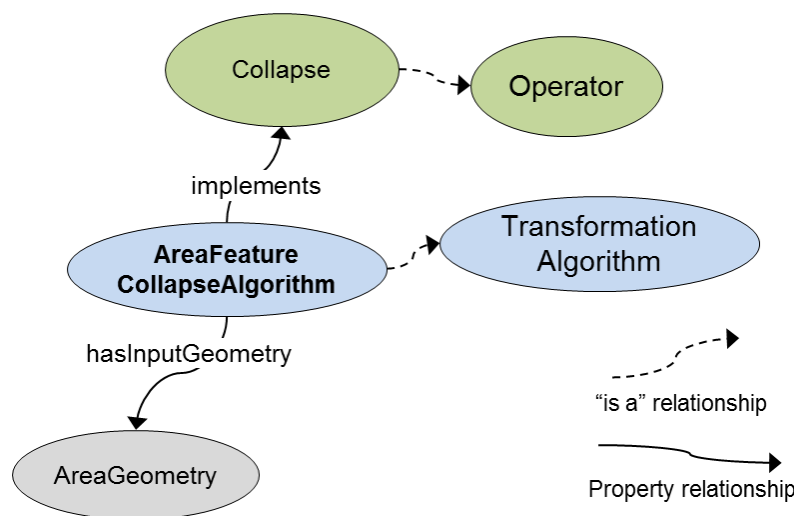


Figure 5.35 Definition of the *AreaFeatureCollapseAlgorithm*

The collapse *operator* can either have line or area input geometry (Figure 5.14) whereas the *AreaCollapseAlgorithm* is limited to area geometry input only (Figure 5.35). A distinct subclass of algorithms that perform area feature collapse as opposed to line feature collapse was created because they are two distinctly different processes.

Figure 5.28 provided the definition required for modelling specialist and non-specialist measure algorithms. That structure can be used as follows. Firstly it is necessary to check whether a specialist algorithm exists for a particular feature type, roads, assuming an area geometry, for example

```
MeasureAlgorithm and measures some HighFeatureDensity and hasInputGeometry
some AreaGeometry and appliesTo only RoadFeatureType
```

The key clause is the final one (in bold) since it limits the return to *only* those measure algorithms that only apply to road features. If no specialist algorithm is found then the next step is to find a general algorithm

```
MeasureAlgorithm and measures some HighFeatureDensity and hasInputGeometry  
some AreaGeometry
```

Ideally the ontology will only return one class and one individual for any query. Otherwise a decision has to be made by the user. For example, the ontology returns *three* operators to remedy congestion in accident features; *SelectionByAttribute*, *Amalgamation* and *Aggregation* which produce the outputs displayed in Figure 5.3c, Figure 5.3e and Figure 5.3f respectively. This is because all three operators have the desired effect but there is not enough information to commend one over the others. Possible solutions to this problem will be discussed in Chapter 7.

By using the Manchester syntax queries, the ontology can be evaluated as it is built and changes made where necessary. When anything other than minor changes are made, the directed graphs should be updated to reflect the changes (Figure 5.1). The graphs can act as a documentation of the ontology.

Another advantage of using Protégé to create the ontology is that the concept of a *defined* class can be used to check whether two concepts are equivalent. A defined class is where the necessary *and* sufficient conditions for membership have been stated. This is in contrast to a *primitive* class where only the *necessary* conditions have been defined. If two operator classes are marked as defined classes then the reasoner will highlight them if they are equivalent. That is, they have the same definition and either one or both definitions need to be refined.

The reasoner can also be used to check the ontology's *consistency* in general, and in particular, for example ensuring that there are no individuals in *unsatisfiable classes*, that is, a poorly defined class that cannot, for logical reasons, have any members.

5.8 Conclusions

The ontology was designed using the methodology described in Chapter 3. To what extent was the methodology successful? The building of the ontology was a long process and large number of versions were created before the competency questions could be answered to satisfaction. Ontology design is "*necessarily an iterative process*" (Noy & McGuinness, 2001, p4), however, the process was at times less iterative than *circular* where some classes and relations were created, discarded and then recreated.

Noy and McGuinness (2001) assert that “*there is no one correct way to model a domain*” (p4). The design of a relational database, in comparison, follows a well-defined process of *normalisation* (Codd, 1970). However, an ontology represents a higher level of abstraction than a relational database and provides more *enriched meaning* (Martinez-Cruz et al., 2012), which makes their design more difficult. It has been suggested that since building ontologies by hand is error prone, they could be derived from existing relational databases (Man et al., 2005). This has been attempted in the geospatial domain but with *material entities* such as buildings (Baglioni et al., 2007) whereas the problems with the development involved defining more *immaterial* concepts such as the *operator* and *geometry*.

A badly-designed methodology might be another cause of ontology design difficulties. The absence of methodologies aimed at specifically designing *application* (or task) ontologies meant that one was developed specifically for the purpose of the research. However, the informal and semi-formal conceptualisation phases were based heavily on the established Noy and McGuinness (2001) methodology so it can be assumed that the methodology is not at fault. In addition the purpose of the ontology was made explicit before design started. It may be that ontology design is by its nature a complex process and its efficiency is dependent on the experience of the designer.

One frequent difficulty in the ontology design was in deciding whether a concept should be represented as a class or as an individual. The problem with the concept of geometry has already been discussed (section 5.4.2). However, the concept of the *operator* represents a variation on this problem. It is clear that the different operators should be represented as subclasses of the operator class since they have different properties from each other and from the superclass. However, the subclasses currently have no individuals and it is hard to conceive what they could be. This is addressed in Chapter 7.

Although the competency questions were answered by the ontology there are still a number of issues that require consideration.

The ontology is not sufficiently refined to suggest a single operator for the resolution of congestion in road accidents (section 5.7). This can be seen by comparing the definitions of the three operators, *SelectionByAttribute* (Figure 5.18), *Amalgamation* (Figure 5.20), and *Aggregation* (Figure 5.25). The definitions are not sufficiently distinct. If the system can not make a choice between one of the three operators then it is not the fully automatic system that was aimed for.

One concept that is not explicitly defined in the ontology is *scale*, which seems incongruous in a domain so concerned with scale. Many algorithms and generalisation workflows are defined in the literature by their optimum source and target scales (Benz & Weibel, 2013; Stoter et al., 2014; Chaudhry & Mackaness, 2005; Revell, 2004). The existence and degree of geometric conditions, such as congestion is, of course, dependent on the scale at which they are mapped. However, for a given condition for a given set of features, the ontology will suggest the same operator(s), whatever the degree of the condition. It would be useful, therefore, to investigate the potential for making the choice of operator scale-dependent, which may lead to more efficient generalisation.

Another option would be to give the user a visual indication of the output of each operator, similar to the diagrams in Figure 5.3 and allow the user to make the choice. However, the conclusion could be that there is a need for a more sophisticated expression of user requirements than merely the features that should be mapped and the map scale and extent.

Some of the challenges specific to the road accident use case were identified and resolved earlier (section 5.5) but the ease at which the ontology can be extended to other use cases and other feature types needs to be examined. Chapter 7 will discuss other use cases and also address the issues raised above.

The differences between different classifications of operators were highlighted in section 2.7. The *informal* natural language descriptions of the existing taxonomies (Foerster et al., 2007a; McMaster & Shea, 1992; Roth et al., 2011) can lead to confusion where it is difficult for humans to determine an appropriate operator, let alone a machine. Some existing operator definitions such as *add* and *eliminate* (Roth et al., 2011) and *enhancement* (Foerster et al., 2007a) are too broad and they have been refined to allow for automatic selection. This was aided by a technique that allows for the creation of *characteristic signatures* for operators, with which operator definitions could be compared and then encapsulated *formally* in an ontology (section 5.4.2). This technique needs to be refined and extended to ensure a more complete definition of operator characteristics.

The reader might conclude that this process has merely created yet another generalisation operator taxonomy to add to the confusion described in section 2.7. However, a taxonomy is concerned with labelling and grouping concepts whereas the ontology is concerned with formalising the characteristics of the concepts and, the labels applied to the concepts are essentially irrelevant.

The ontology has been evaluated by phrasing the competency as Manchester Syntax queries. However, there are a number of unresolved issues and the true test of an application ontology is in the successful development of an application that uses the ontology. The next chapter describes the development of a prototype on-demand mapping system that will use the ontology.

6 Implementing the on-demand mapping system

6.1 Introduction

Following the creation of the ontology the next stage is to implement a prototype on-demand mapping system to evaluate it. As the ontology is an *application* ontology the development of the prototype is seen as an important stage in the evaluation phase of the ontology building methodology (section 3.7.3). The prototype will be limited to implementing the road accident use case.

Chapter 4 described a methodology for building an on-demand mapping system in an implementation-independent manner, based on the adapted CommonKADS methodology (Schreiber et al., 2000). This chapter describes the *platform* design phase, which is the implementation of the *architecture* design of Chapter 4 (Figure 4.17) into software. The ontology contains the *domain* knowledge and chapter 4 described the *control* knowledge in terms of *task* knowledge and *inference* knowledge. The prototype will use the task knowledge to orchestrate the inference knowledge, which will utilise the domain knowledge.

This chapter will also describe the *procedural* knowledge that is held by the algorithms developed to implement the use case. The *measure* algorithms will be used to derive the *Degree of Generalisation*, which will be used as a parameter to the *transformation* algorithms.

6.2 Data

The area covered is the centres of the adjacent cities of Manchester and Salford in Greater Manchester (Figure 6.1). This area was selected since it provides a variety of roads including motorways, major radial and ring roads, city centre, and residential roads. The extent is limited because the algorithms developed were not optimised and were limited by the number of features they could process.

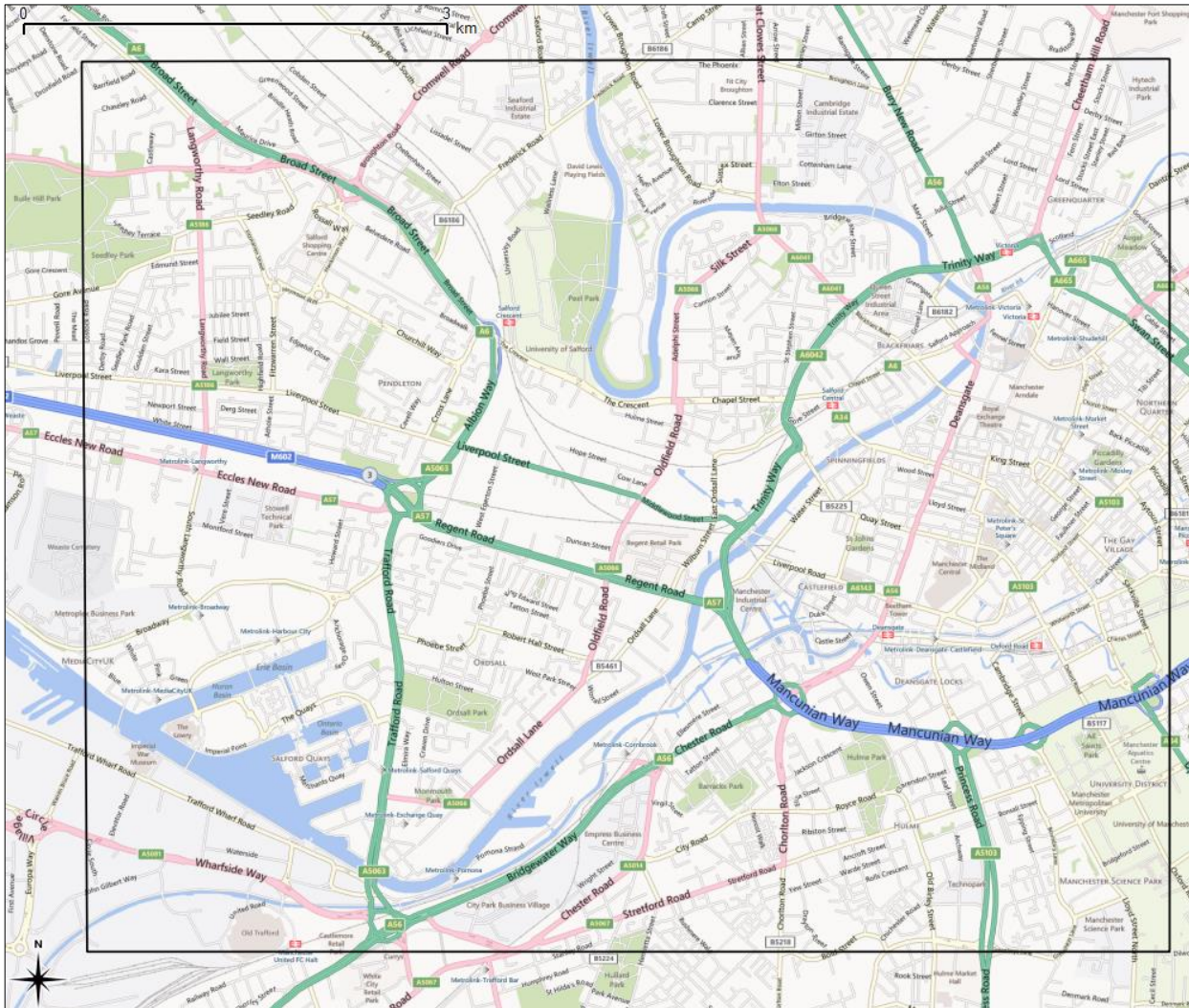


Figure 6.1 Extent of use case data (background map © Microsoft Corporation 2014)

6.2.1 Accidents

The dataset consists of 9306 road accidents from 1994 to 2008. The data is provided by Transport for Greater Manchester via the DataGM open data website¹¹. The data only consists of those accidents that occurred on public roads, were reported to the police, and where at least one person was injured. The data was stored using the ESRI *Shape* file format.

6.2.2 Roads

The road data is provided by the Ordnance Survey via the Digimap Service (University of Edinburgh, 2014). Two road datasets were used; *MasterMap*, a polygon-based topographic dataset and the *Integrated Transport Network* (ITN) described by lines. The data was converted from Geographic Markup Language (GML) format to the Shape file format.

¹¹ www.datagm.org.uk

References to the accident dataset and the two road datasets were added to the ontology as individuals in subclasses of the *feature collection* class. The properties of each individual such as the filename can also be defined in the ontology at this stage.

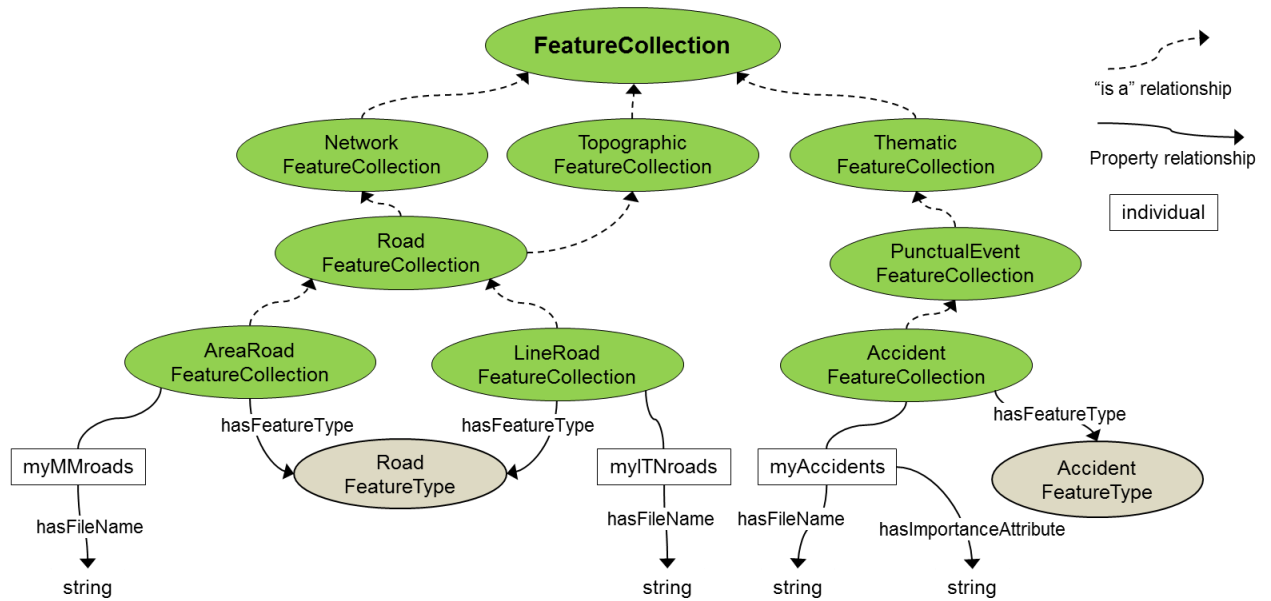


Figure 6.2 Defining feature collection individuals

This is contrast to the design proposed in Figure 5.10, which has the concepts of thematic and topographic features associated with the *feature type* class. The decision to have a more enriched feature collection definition was made because the feature collection is a more important concept than the feature type; a result of the aim to classify particular mapped features rather than general types of features. This is an example of why the implementation of the ontology is an important part of its evaluation and how it can influence its design.

The following two sections describe the algorithms developed for the prototype. The algorithms have not been optimised and there may well be better implementations but the intention is not to develop a set of ideal measure and transformation algorithms but to test the concept of using an ontology to automatically select algorithms.

6.3 The measure algorithms – identifying problem features

The measure algorithms will be used to answer the *when to generalise* question of the McMaster and Shea (1992) model. For the use case the amount of *congestion* in the accident and road features need to be determined.

6.3.1 Measuring congestion in accident features

Congestion is measured by identifying clusters in the *mapped feature collection* (Figure 4.13). The DBSCAN point clustering algorithm (Ester et al., 1996) was used since it is a density-

based algorithm that will eliminate noise points, unlike other algorithms such as K-means (MacQueen, 1966) that add every point to a cluster, which will not help in the identification of hot-spots. The algorithm has two parameters, the values for which should be derived automatically. *MinPts*, the minimum number of points in a cluster can be set to 4 (Ester et al., 1996). *Eps*, a real world distance value which is used to determine if two points are in the same cluster, requires more consideration since a suitable value is dependent on scale. The *Eps* value defines how close features have to be before they are considered congested and acts as a form of threshold. If two or more symbols overlap on a map then their real world separation could be measured in centimetres or kilometres depending on the scale but for the sake of legibility they need to be generalised. Therefore a scale-based equation to derive *Eps* was developed (Equation 6.1).

$$Eps = \left(\frac{symbolSize}{scale} \right) \times FeatureWeight$$

Equation 6.1

The application was developed using *GeoTools*, an open source GIS Java toolkit (GeoTools, 2014) and it is the GeoTools definition of scale that has been used

the number of pixels per Coordinate Reference System (CRS) distance units.

In effect, this is the number of pixels on the screen per metre on the ground. As scale decreases the value of *Eps* increases, i.e. features that were too far apart to be congested appear congested at a smaller scale (Figure 6.3a and Figure 6.3b). This assumes a fixed symbol size (in pixels) whatever the scale. In the same way the *Eps* value decreases with decreasing symbol size. Given a small symbol size there might be no need for generalisation (Figure 6.3c). The symbol size in this implementation does not automatically change with scale and could be chosen by the user.

The *featureWeight* is a unit-less multiplier dependent on the nature of the source data and will be discussed in detail later.

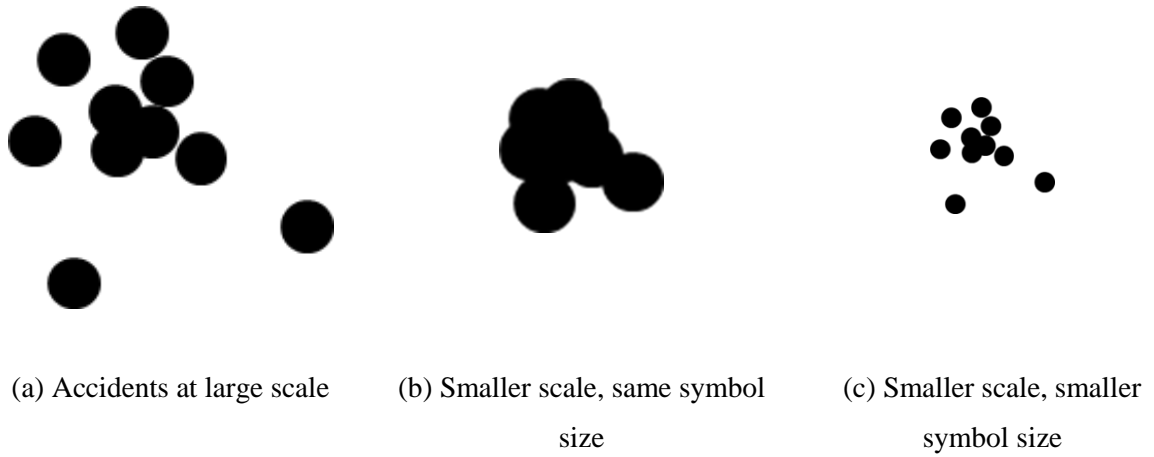


Figure 6.3 Accidents represented as points at different scales and symbol sizes

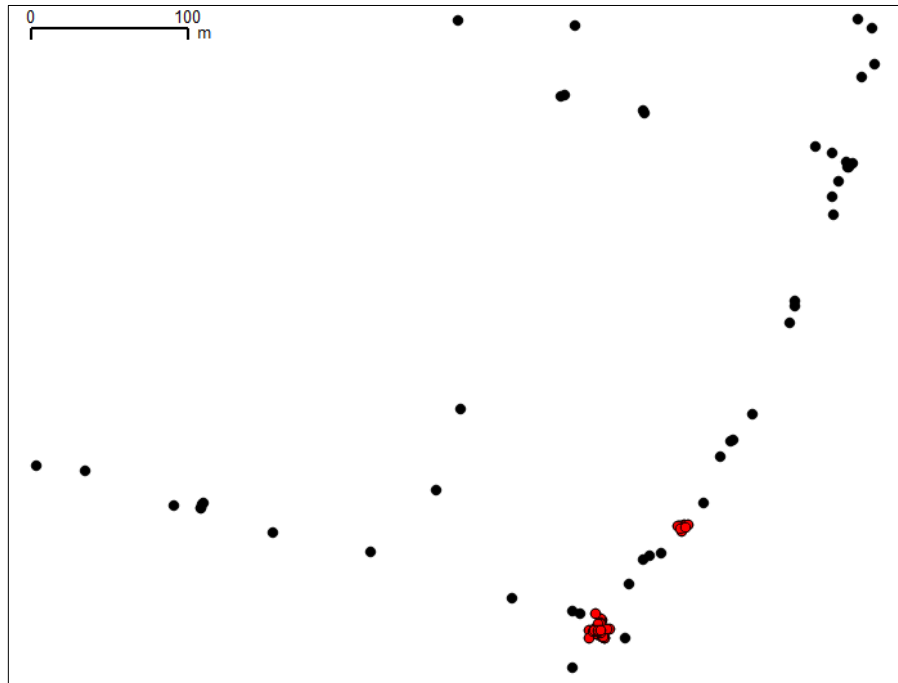
The measure of high density in point features is implemented by a generic algorithm, *findHighPointDensityClusters* (Table 6.1). It works on the assumption that any cluster is an area of high density and rather than return a measure of density the algorithm returns a set of (problem) feature collections, each one representing a cluster (Figure 6.4).

Input parameters		
Name	Data type	Description
MappedFeatureCollection	SimpleFeatureCollection	Mapped features
PointSize	Integer	Size of the feature symbol
Scale	Double	Number of pixels per CRS distance units
FeatureWeight	Double	Feature related weighting
Output		
	SimpleFeatureCollection set	Set of simple feature collections, each collection representing a cluster of problem features

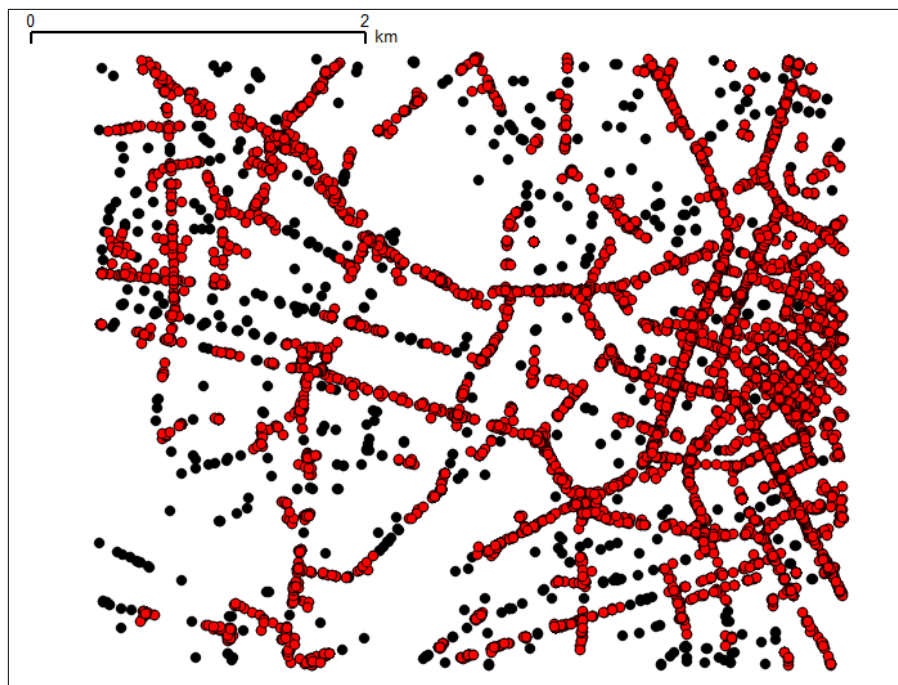
Table 6.1 Parameters for the *findHighPointDensityClusters* measure algorithm

The algorithm first calculates the Eps parameter value and then calls the DBSCAN algorithm, which was implemented using the version developed by Tan et al. (2006). The *implementation* of the algorithm is represented in the ontology as an individual in the class *PointFeatureDensityMeasureAlgorithm* (Figure 5.28). The individual holds a reference to the Java function implementing the algorithm.

At a large scale (Figure 6.4a) only two clusters have been identified in this example, but at a smaller scale the number of clusters increases and the number of features in each cluster increases, as features that were not previously in clusters join a cluster since *Eps* increases with decreasing scale (Figure 6.4b).



(a) Large scale



(b) Small scale

Figure 6.4 Problem accident features (highlighted in red)

6.3.2 Measuring congestion in road segment features

The road segments can exist as area or line features, therefore two different algorithms are required to measure the feature density. As with the point density algorithm both algorithms return a set of problem feature collections, each collection containing the features in a cluster, rather than a measure of density. The density of crossroads is used as a measure of road feature density¹² (Figure 6.5 and Figure 6.8).

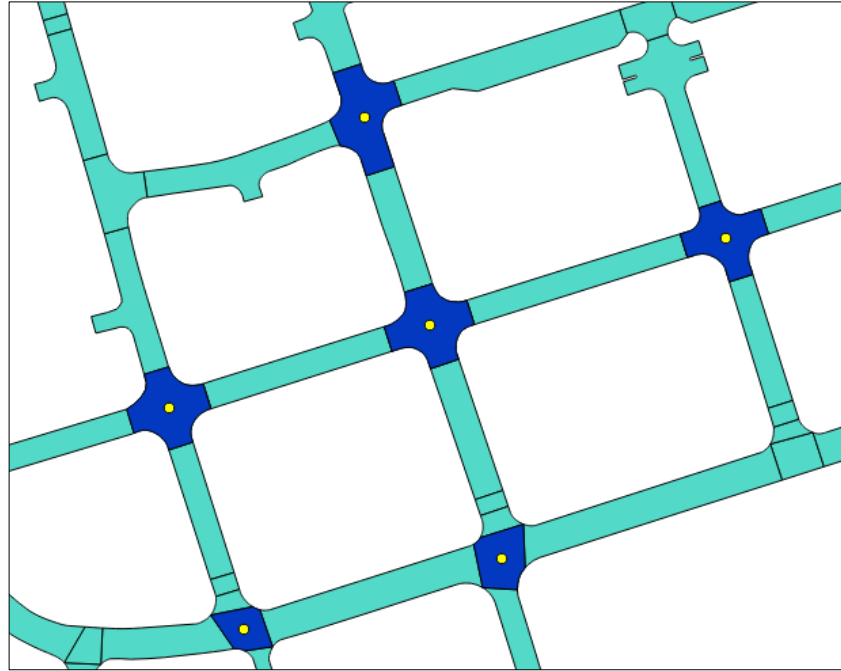


Figure 6.5 Identifying high density of road features as *areas* using crossroad density

For road features represented as areas (Figure 6.5) the crossroad features were identified in a pre-processing stage by identifying those features that shared a border with three or more features. The measure algorithm (*findHighDensityAreaCrossroadClusters*) then identifies the centroids of the crossroads features and uses the implementation of DBSCAN to identify clusters of these points. However, in this case Eps is calculated differently from that method used for point features (Equation 6.1) by omitting the *symbol size* component (Equation 6.2). This is because there is no obvious equivalent for area features to the *symbol size* in point features. Scale and featureWeight are again input parameters to the algorithm (Table 6.2).

$$Eps = \left(\frac{1}{scale} \right) \times featureWeight$$

Equation 6.2

¹² The density of all types of road junctions, not just crossroads, could also be considered.

Once the clusters of crossroad centroids were identified, it was then necessary to identify those road segments that were related to the cluster. Firstly the centroid of each cluster is determined by calculating the mean X and Y values of the cluster points. This centroid is used as the centroid of a circular buffer. The radius of the buffer is determined by the spread of the crossroad centroids and the radius is set to $\frac{1}{2}$ of the largest dimension of the bounding box (Figure 6.6a). Any road segment that intersects with the buffer is then marked as part of a cluster of road segments (Figure 6.6b). The algorithm returns a set of problem feature collections, each collection containing the road segments in a cluster. An example of congested road features is shown in Figure 6.7.

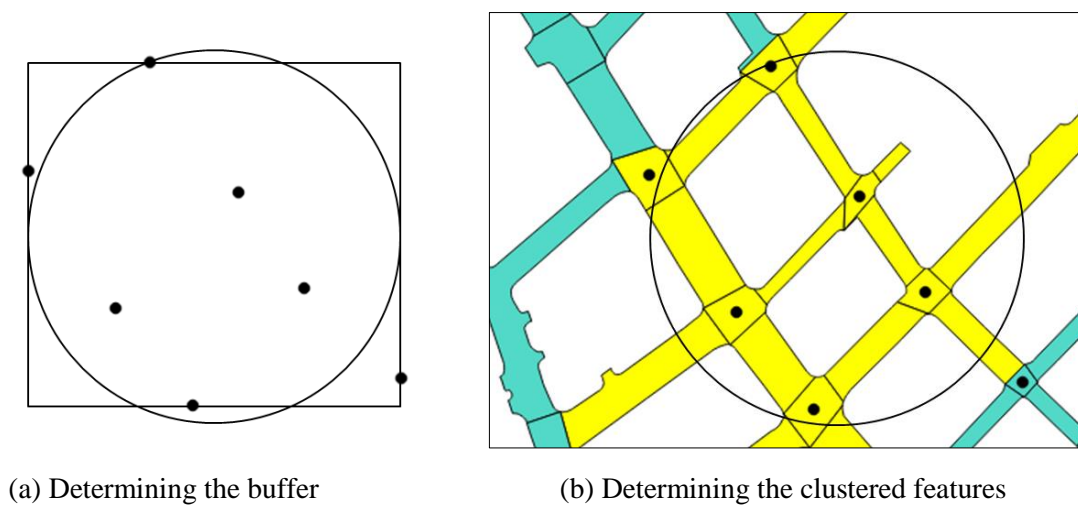


Figure 6.6 Determining a high density cluster of road segments

Input parameters		
Name	Data type	Description
MappedFeatureCollection	SimpleFeatureCollection	Mapped features
CRS	CoordinateReferenceSystem	CRS of mapped features
Scale	Double	Number of pixels per CRS distance units
FeatureWeight	Double	Feature related weighting
Output		
	SimpleFeatureCollection set	Set of simple feature collections, each collection representing a cluster of the mapped features

Table 6.2 Parameters for the *findHighDensityAreaCrossroadClusters*

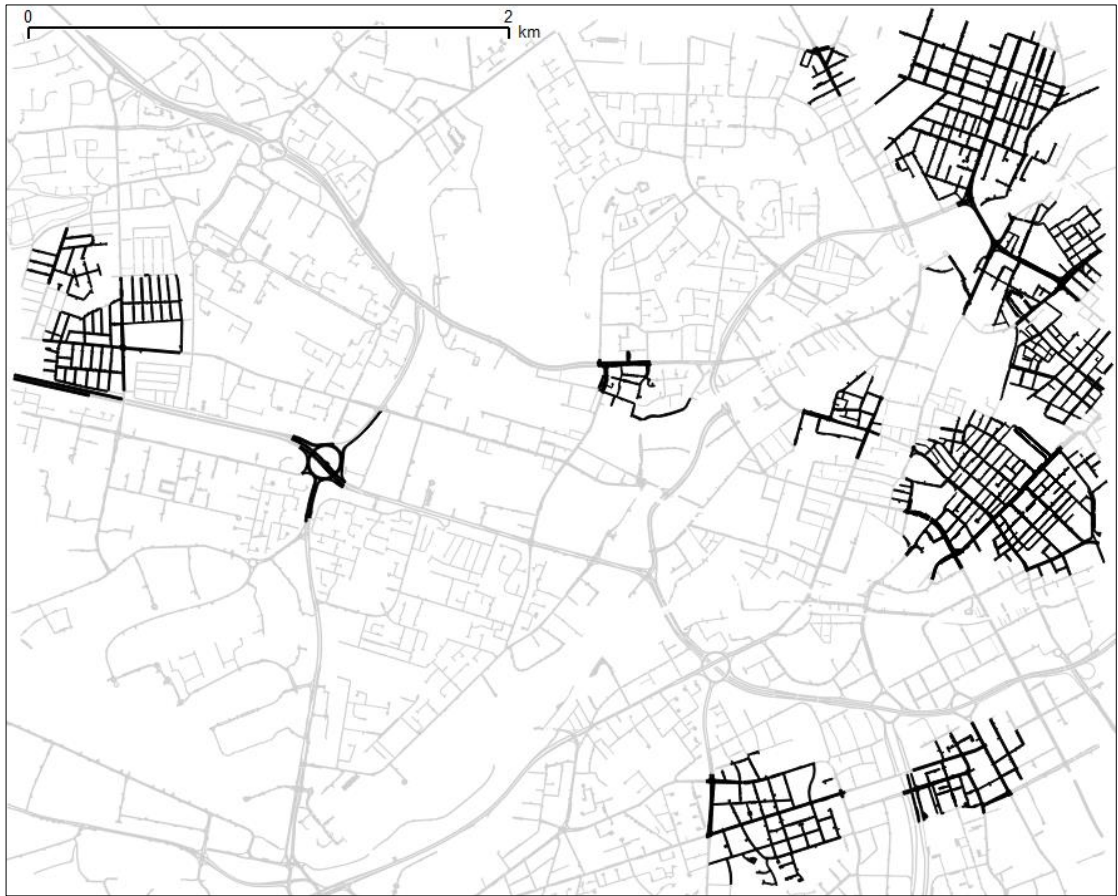


Figure 6.7 High density area road segments (highlighted)

The density of crossroads was also used to identify clusters of congested road features when they are represented as lines (Figure 6.8). In contrast to the area feature density measure algorithm, no pre-processing was required. Each feature in the source dataset has a start and end node identifier and the algorithm (*findHighDensityLineCrossroadClusters*) identifies each unique node in the dataset and then determines the number of links associated with that node. If the number of links is greater than 3 then that node is flagged as a crossroad and a temporary set of point features is created (Figure 6.8). Clusters are again identified using DBSCAN, but this in this case *Eps* is calculated using Equation 6.1 (p112) and the symbol size is the width of the lines used to represent road links (all roads are mapped with the same width). The remainder of the process is exactly the same as for area road features and the output is the same; a set of problem road feature collections, each collection representing a cluster. The parameters are the same as for the previous algorithm (Table 6.2) with the addition of a symbol size parameter, representing the road feature width.

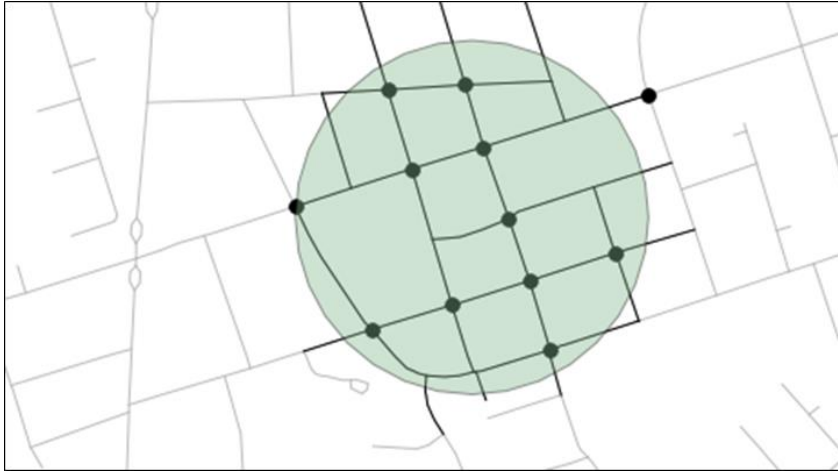


Figure 6.8 Identifying high density of road features represented as *lines* using crossroad density

Two different values of the *featureWeight* (Equation 6.1 and Equation 6.2) were used, one for accidents and one for roads. For a given dataset the default *featureWeight* is fixed for all scales. When identifying clusters of accidents, values of around 3 were found to be appropriate using a subjective visual assessment. However, a *featureWeight* of 3 for the roads results in too few features being identified as being part of a cluster and therefore a congested map. This is because the points represent only the crossroads in the mapped FC and not the actual features. Therefore a higher weight factor, and thus a higher *Eps*, is more suitable for the road features. After some experimentation, a value of 25 was found to be more appropriate. These values were determined by a visual estimate of what gave acceptable results, but this is obviously subjective and dependent on the user. What are not subjective are the measurement of scale and the symbol size. The *featureWeight* therefore allows for some user discretion and could be changed from the default by the user in a more sophisticated implementation.

6.4 Calculating the Degree of Generalisation

For both feature types, once the problem features have been identified by the measure algorithms, the Degree of Generalisation (section 4.4.4) can be calculated (Equation 4.1).

The value is then standardised to an integer between 1 and 9, where 9 indicates a maximum degree of generalisation. Integers were used because using anything else would give an incorrect impression of exactness in the process. The value 9 is used as an upper limit since the value 10 implies a value of 100% generalisation, which in turn implies the removal of all features. Although the intention is for automatic generalisation, where the user is unaware of the underlying processing and parameter values, it will be easier to allow for some user interactivity if a simple range of values for DoG is used. This enables interactions such as

“the accidents were generalised with a degree of generalisation of 6, would you like to repeat with a different value?” Once calculated, the DegreeOfGeneralisation can then be passed to the appropriate generalisation algorithm.

6.5 Algorithms for generalising the accidents

This section describes three algorithms for generalising the road accidents. It was necessary to develop new algorithms in order to utilise the DegreeOfGeneralisation concept.

All three algorithms use the DoG to determine the number of features to retain (Equation 6.3). A DoG of 1 will generate a target of removing 10% of mapped features and a maximum value of 9 will generate a target of 90%.

$$\text{numberOfFeaturesToRetain} = \left(1 - \frac{\text{DegreeOfGeneralisation}}{10}\right) \times \text{totalNumberOfFeatures}$$

Equation 6.3

6.5.1 Selection by attribute

One way to solve the problem of point congestion is to select only the most important features. This has been termed as the *selection by attribute* operator in the ontology (section 5.5.1). To apply this operator the source data will require an attribute that can be used to rank features by importance. This attribute is defined in the ontology (Figure 6.2) and its name is passed as a parameter to the algorithm (Table 6.3). In the case of road accidents a *severity* attribute can be used, where the value 1 represents a fatal accident, 2 a severe accident and 3 a slight accident.

The algorithm then uses this attribute and the number of features to retain (derived from the DoG) to determine which features to retain. For example, if the target number of features to retain was 1000 then the algorithm would retain features of severity value of 1 and 2 (Figure 6.9) since that would return a number closest to the target. A disadvantage of selection by attribute is that it applies to the complete mapped FC and some features that were not identified as problem features may still be removed.

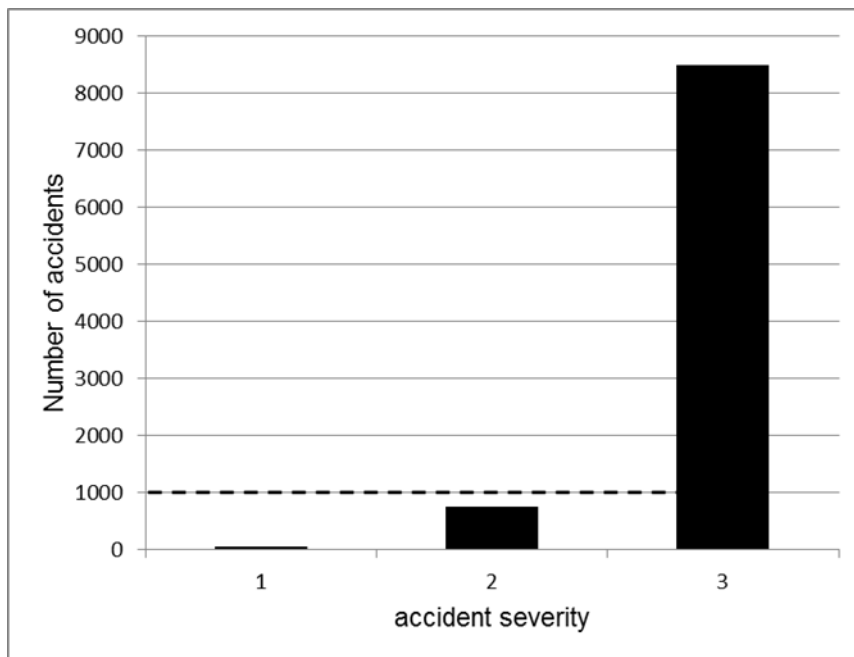


Figure 6.9 Using the DoG to determine the class of features to retain

Input parameters		
Name	Data type	Description
MappedFeatureCollection	SimpleFeatureCollection	Mapped features
ImportanceAttribute	String	Name of the importance attribute in the dataset (extracted from the ontology)
DegreeOfGeneralisation	Integer	Degree of generalisation calculated from the measure algorithm results
Output		
	SimpleFeatureCollection	Simple feature collection containing the retained (accident) features

Table 6.3 Parameters for the select by attribute algorithm

For useful results, this technique is dependent on there being an even distribution of features and a reasonable range of values in the importance attribute. The road accident data has only three values for severity and is heavily weighted to the *slight* accidents category (Figure 6.9). The result is a lack of resolution for different DoG values as can be seen in Figure 6.10. There is only a close match between the target number of features retained and the actual number of features retained when the DoG is relatively low and relatively high and there is no actual removal of features until the DoG is 5 and over. This, however, is more a limitation of

the selection by attribute operator and the source data rather than the DoG concept. If, for example, the *year* of each accident was used as an importance attribute, where more recent accidents were given priority then there is a larger number of values (years) and a more even distribution of the data (compare Figure 6.11 to Figure 6.9) and there will be a variation of output from the algorithm over the range of DoG values.

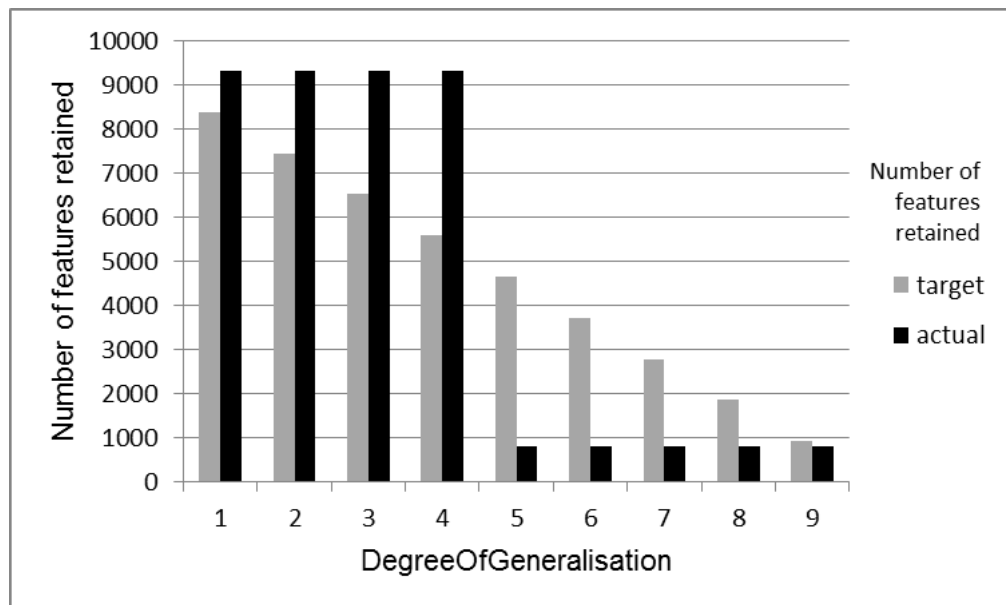


Figure 6.10 Target and actual features retained from the full accident dataset with selection by attribute

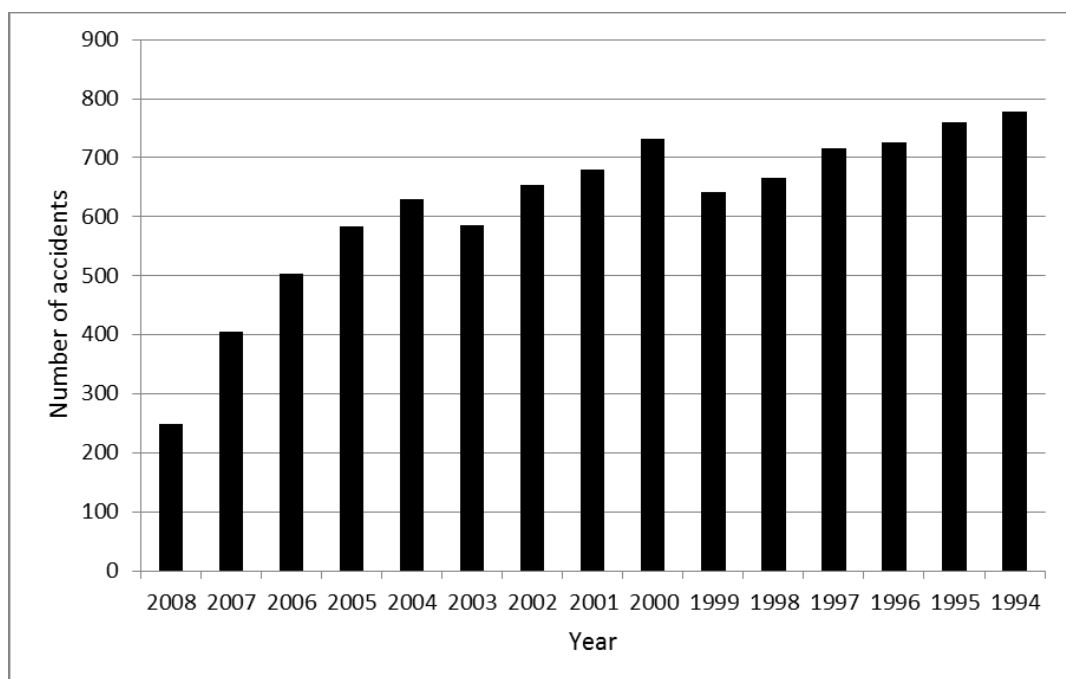


Figure 6.11 Number of accidents in each year of the sample dataset

6.5.2 Amalgamation of accidents

This algorithm aims to implement the amalgamation operator (section 5.5.2). The input to this algorithm differs from the previous one because, rather than take the full set of mapped features as input, it takes only the clusters of high density features identified by the measure algorithm *findHighPointDensityClusters* (Table 6.4). The number of features to retain is again derived from the DoG (Equation 6.3, p119). The algorithm orders the input clusters by descending size (in terms of number of members) and then, starting from the largest, adds them to a set of clusters until the target number of features is reached. The resulting set of clusters is turned into a set of polygon features by drawing a convex hull around each cluster (Figure 6.12).

Input parameters		
Name	Data type	Description
ClusterSet	Set of SimpleFeatureCollection	Clusters of point features identified by the measure algorithm
CRS	CoordinateReferenceSystem	CRS of mapped features
TotalFeatures	Integer	total number of mapped features
DegreeOfGeneralisation	Integer	Degree of generalisation calculated from the measure algorithm results
Output		
	SimpleFeatureCollection	Simple feature collection of polygons representing the retained (accident) features

Table 6.4 Parameters for the point amalgamation algorithm

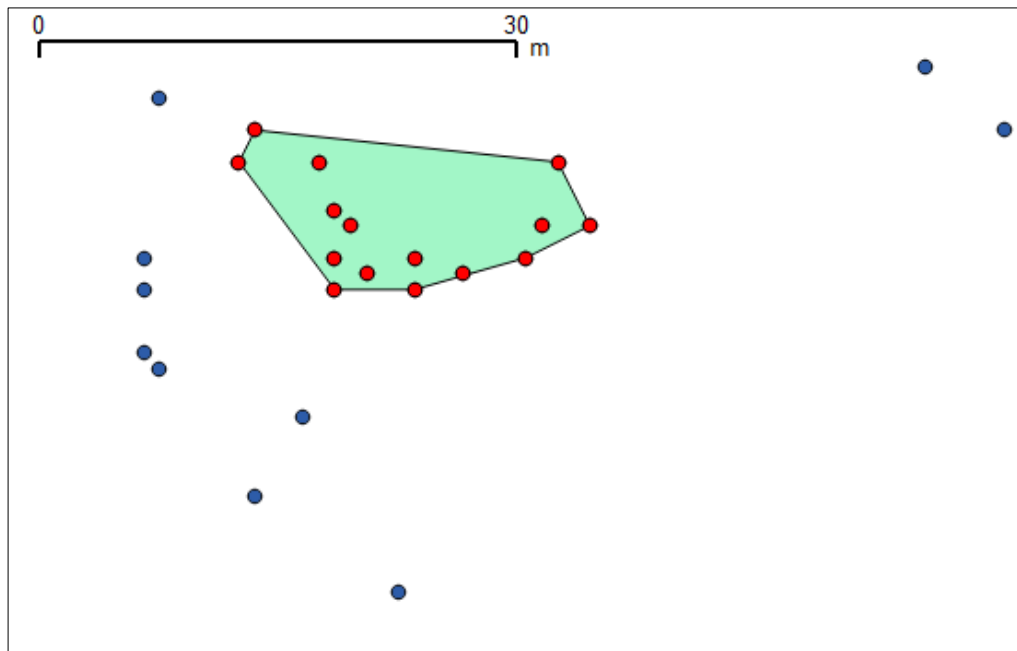


Figure 6.12 Amalgamated accident cluster with DoG of 6 (accidents in red are those identified as problem features).

Figure 6.13 depicts the generalised accidents (as polygons) and the source data as a reference, mapped at the full extent of the source dataset using the automatically derived value of DoG of 9. It can be seen that only the numerically largest clusters are retained and many congested features will not be represented. If a lower DoG is manually selected then the result is more representative (Figure 6.14). It can be argued that simply converting *all* identified clusters into convex hulls provides the most representative result (Figure 6.15). However, the DoG concept allows for an element of control by the user but it still requires a more sophisticated implementation in this algorithm; when calculating the number of features to retain the algorithm does not allow for the changed representation of those features.

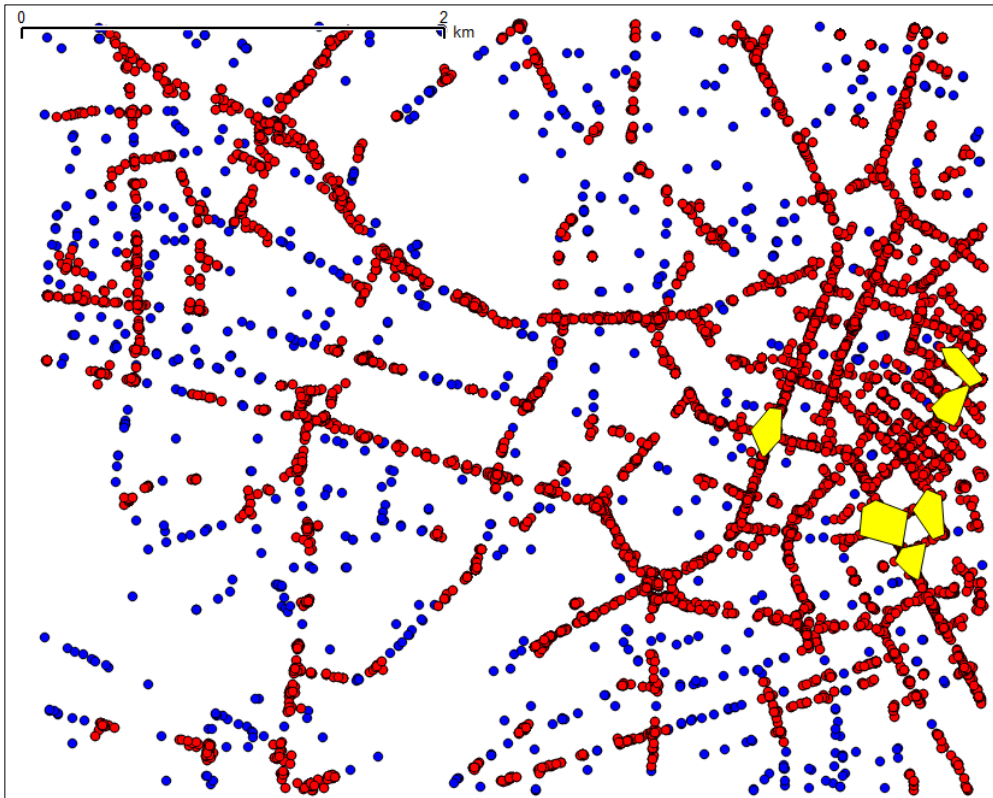


Figure 6.13 Accidents at scale 0.17 pixels/m amalgamated (yellow) with DoG of 9 (problem features in red)

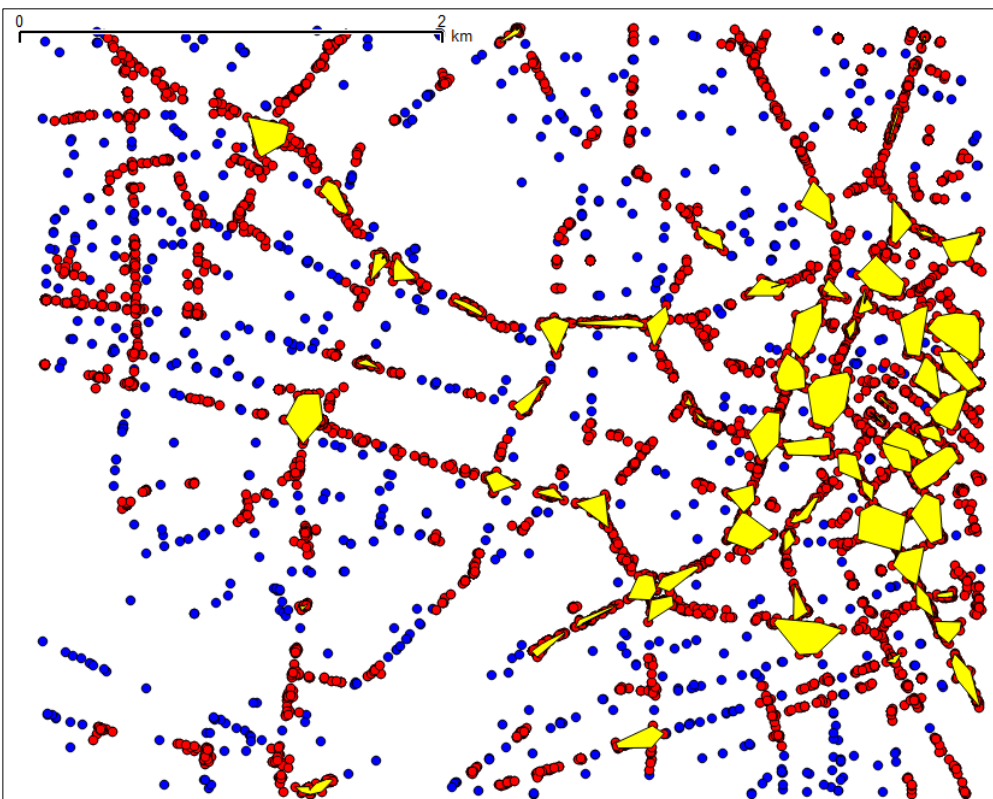


Figure 6.14 Accidents at scale 0.17 pixels/m amalgamated with DoG of 5 (problem features in red)

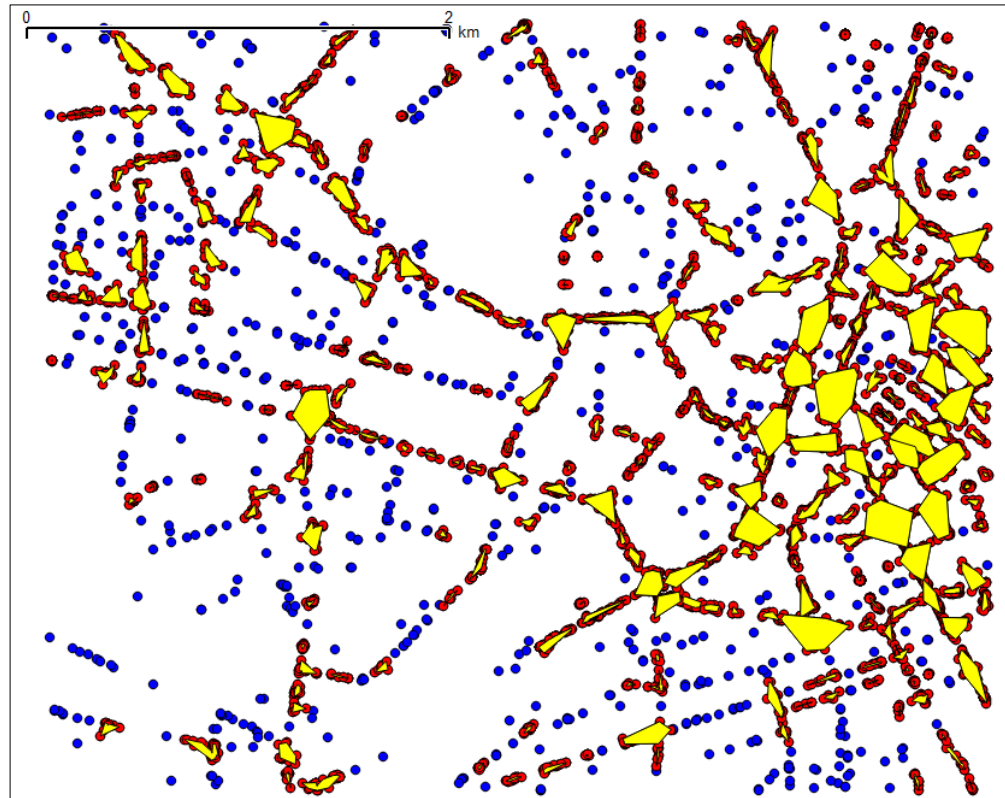


Figure 6.15 Accidents at scale 0.17 pixels/m. All problem feature collections amalgamated

6.5.3 Aggregation of accidents

This algorithm seeks to represent each cluster of accidents as a single point, where the diameter of the point is in proportion to the number of points in the cluster. The operator is termed *aggregation* in the ontology (section 5.5.3). The algorithm has the same input parameters as the amalgamation algorithm (Table 6.4).

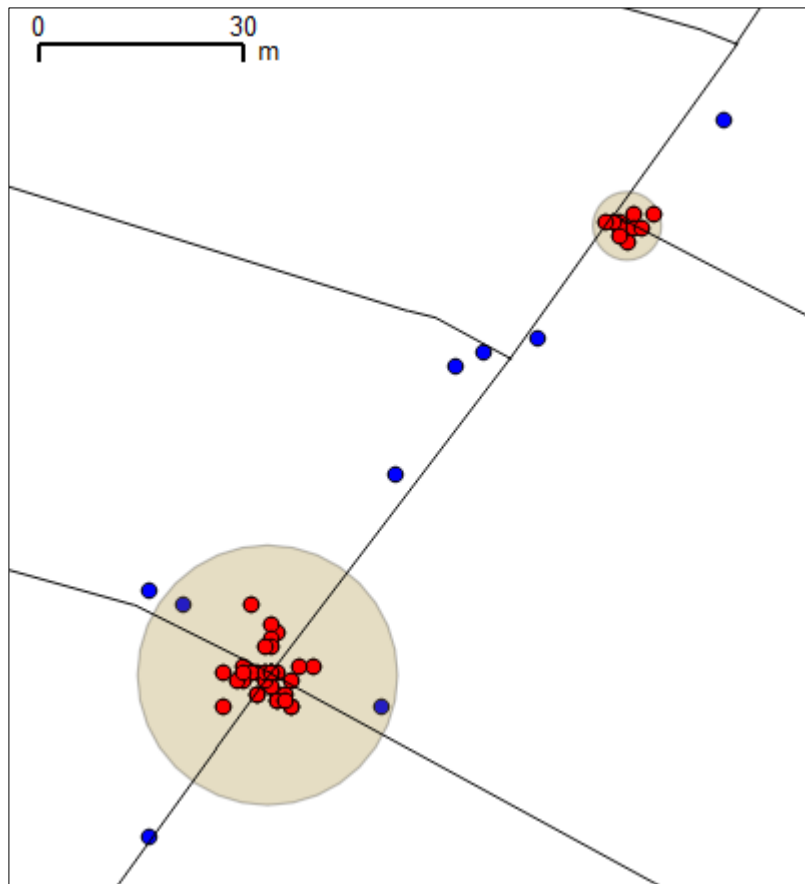


Figure 6.16 Aggregated road accidents (problem features in red)

The result is the same number of features as the amalgamation algorithm but with the clusters represented as points instead of a convex hull (Figure 6.16). The new features do not correctly represent the geographical extent of the cluster, only its numerical size. This can be seen from the point feature representing the cluster at the bottom of Figure 6.16, which overlaps two non-clustered accidents. However, it could be argued that, aesthetically, aggregation is an improvement on amalgamation if Figure 6.17 is compared to Figure 6.18. More importantly aggregation gives a more accurate representation of the total number of features in a hot-spot than amalgamation does.

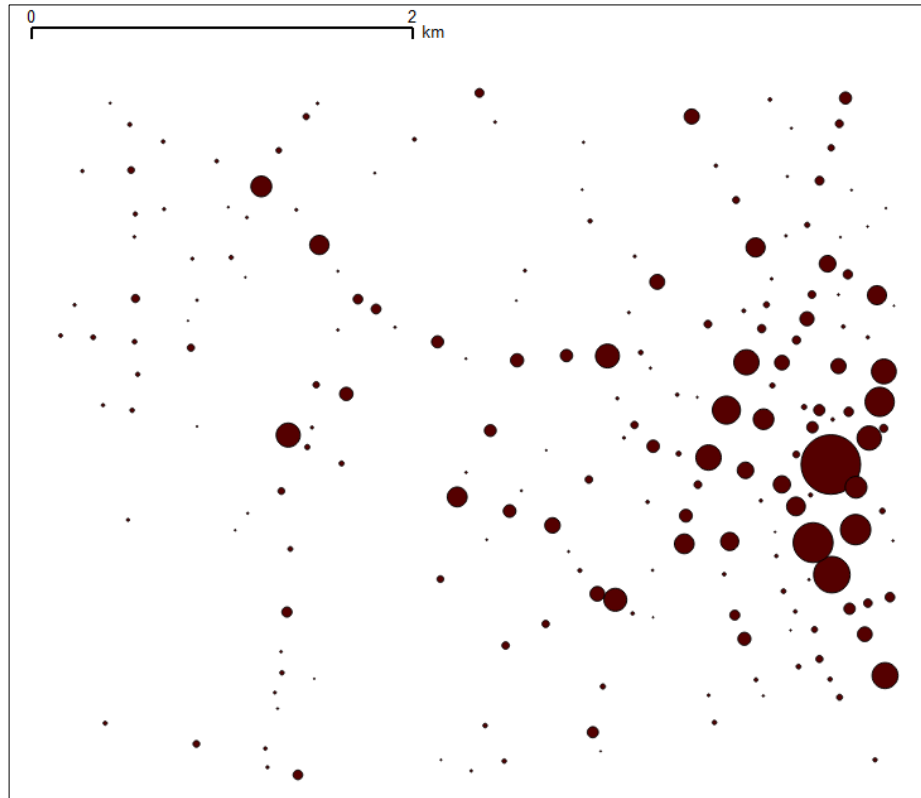


Figure 6.17 Aggregation with a DoG of 1 at scale 0.17 pixels/m

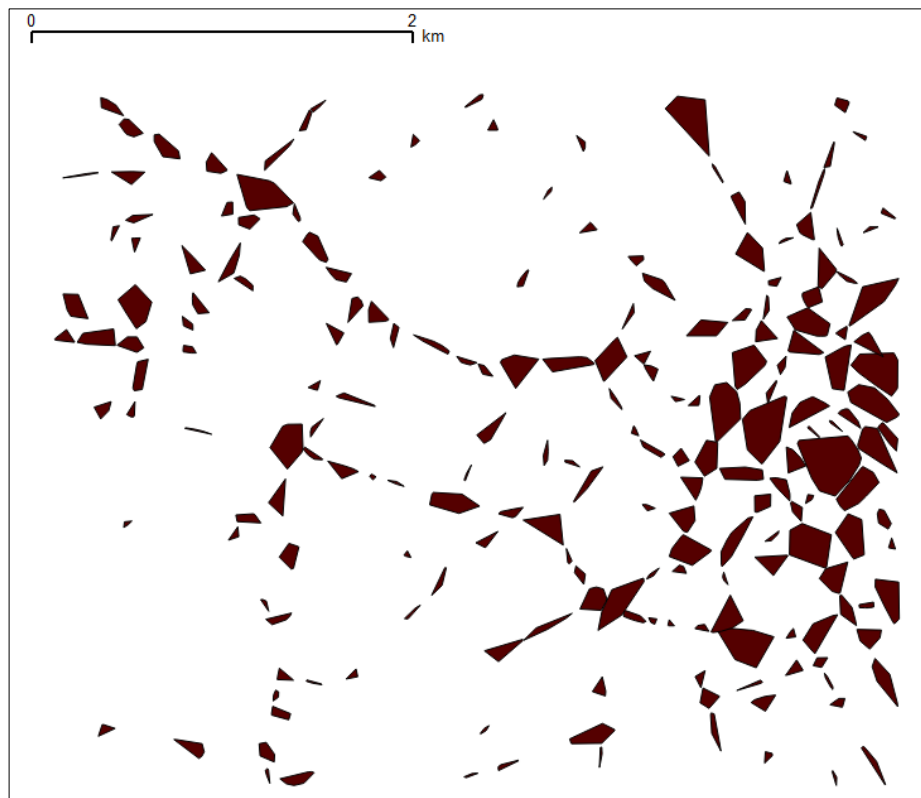


Figure 6.18 Amalgamation with a DoG of 1 at scale 0.17 pixels/m

As with the amalgamation algorithm, the aggregation algorithm does not retain any features that are not in a cluster; generalisation is not restricted to those areas of the map that have

congestion problems. The decision to omit uncongested accidents is not a technical one; they can be easily retained. The equivalent is the retention of isolated buildings, in rural areas for example, where buildings in built-up areas have been amalgamated.

It might be argued that a spatial analysis such as a *kernel density estimation* (Silverman, 1986) could be used to visualise the density of the accident features (Figure 6.19). However, such an analysis produces a raster image and there is no representation of the clusters as features. The latter allows for further analysis; for example, attaching a “*number of accidents*” attribute to road junctions where a road junction intersects a hot-spot. However, it may be useful to investigate whether the kernel density approach could be used to *identify* clusters.

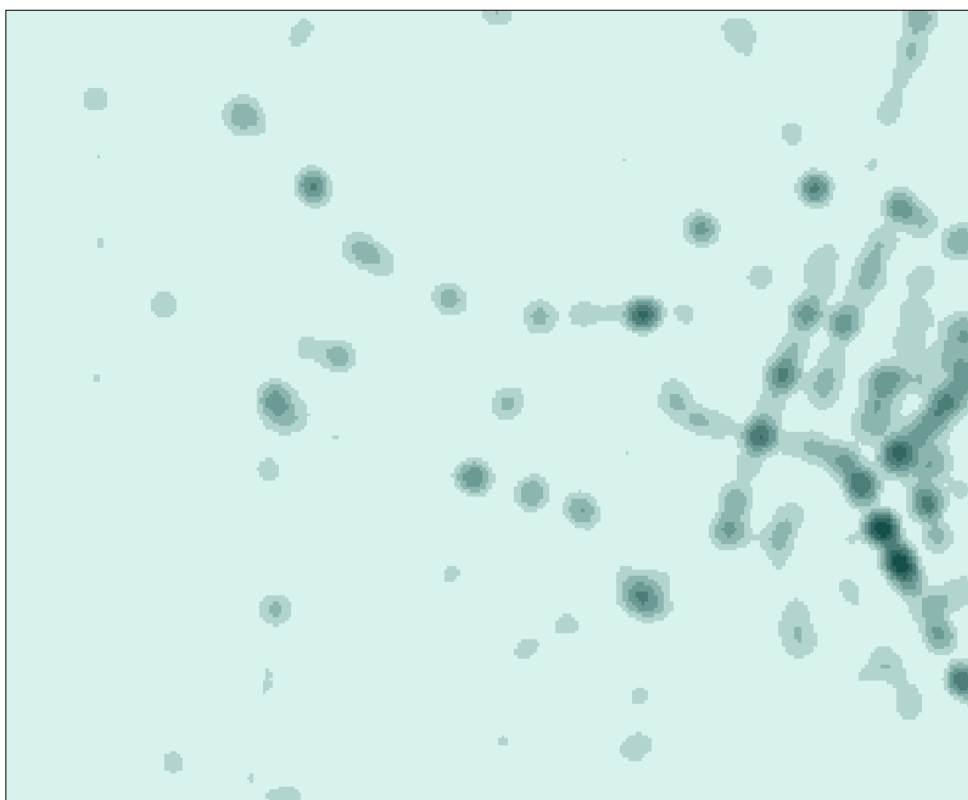


Figure 6.19 Kernel density analysis of the complete set of road accidents (using the ArcMap 10.1 function)

6.6 Algorithms for generalising the road network

6.6.1 Collapse

The collapse operator is a binary operator, there is no concept of a degree of collapse and the DoG concept is not applicable to this operator. Therefore, for the sake of simplicity, the collapse of the MasterMap road network from a set of area features was simulated by simply substituting the area features with the line features of the ITN road network (Figure 6.20).

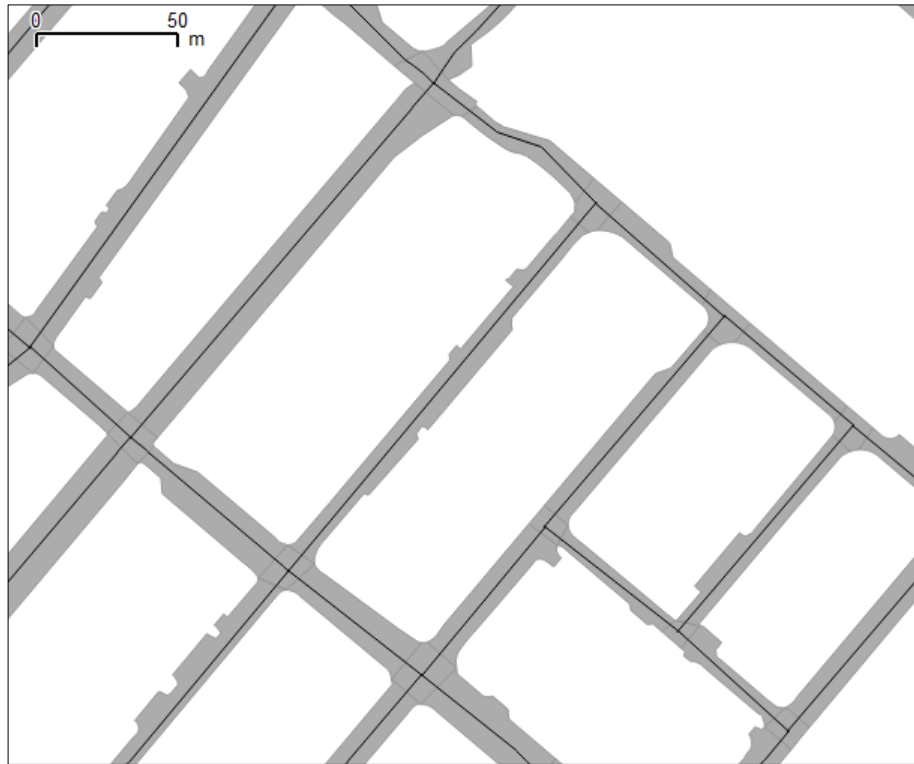


Figure 6.20 Simulating the collapse of area road features using the ITN network (lines)

6.6.2 Pruning the road network

The road *pruning* (or thinning or selection) operator (section 5.5.4) was implemented by a simple application of the *strokes* technique, where road segments are concatenated into chains or strokes based on the *good continuation principle* (Thomson & Richardson, 1999). The strokes method is a well-known generalisation technique and Zhou and Li (2012) have carried out a comparative study of a number of implementations of the method. A more recent approach is the mesh-based approach (Chen et al., 2009), which considers the density of meshes of road segments, where a mesh is a closed region that is bounded by a number of road segments. Even more recently, combined mesh and stroke approaches have been developed (Li & Zhou, 2012) and refined (Benz & Weibel, 2013). Whatever the technique employed, many road selection algorithms have been optimised for a particular target scale (Benz & Weibel, 2013; Weiss & Weibel, 2013; Revell et al., 2005), which is not ideal for on-demand mapping, where the user should be able to select an arbitrary scale. The combined stroke and mesh approach of Li and Zhou (2012) is multi-scale but still has four parameters, two of which are always known but the other two need to be determined.

This implementation of the strokes method simply considers the angle between adjacent road segments to determine whether the segments should be in the same stroke. For example, the links *a*, *b* and *c* in Figure 6.21 are part of the same stroke; as are links *m* and *n*. Link *f* is not

part of the same stroke as *a* and *b* since the angle between it and link *b* is greater than the threshold (the *limiting angle*). The limiting angle is a parameter to the pruning algorithm and was set, after some experimentation, at 20° for this research.

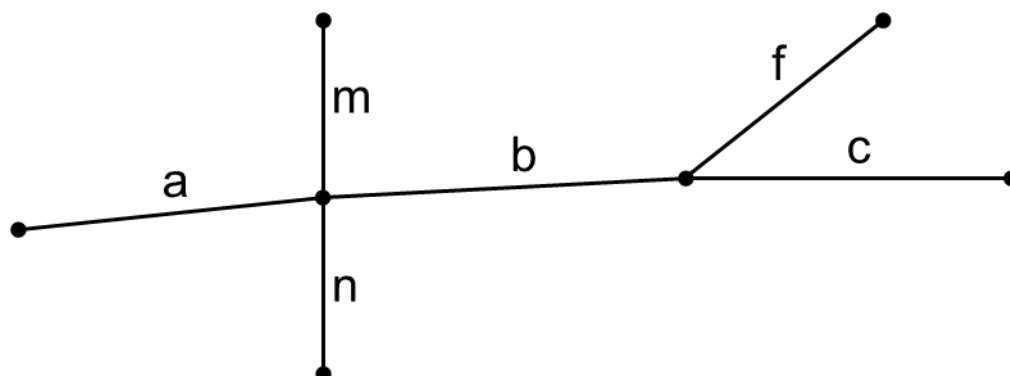


Figure 6.21 Creating strokes using the good continuation principle

A more sophisticated algorithm might also consider the road name and/or its class when determining whether adjacent links are in the same stroke (Zhou & Li, 2012) but the purpose of developing a new algorithm was to simply to test the Degree of Generalisation concept, not to develop the optimal road selection algorithm. An example of some of the strokes generated can be seen in Figure 6.22. Those road sections in the same stroke are labelled with the same numeric identifier.

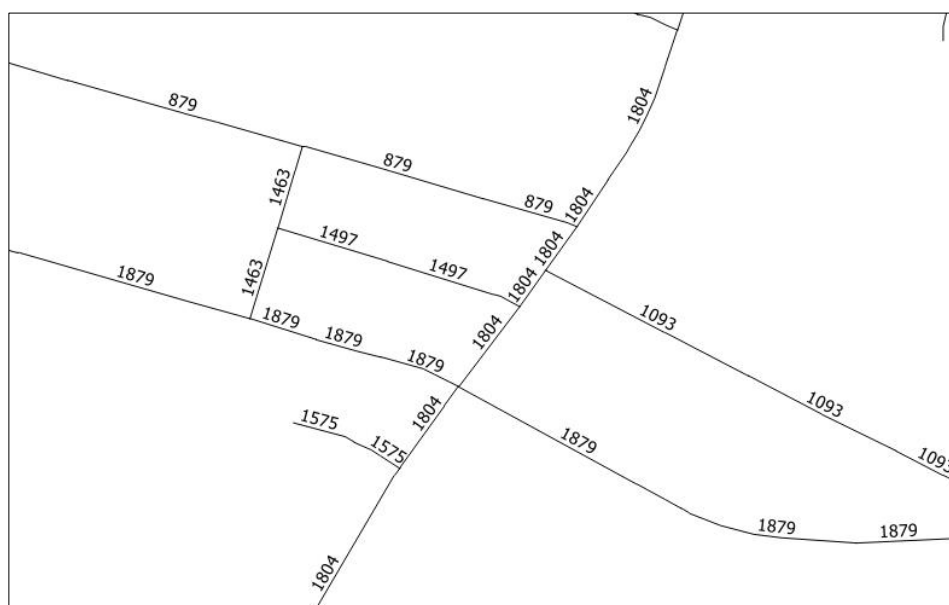


Figure 6.22 Strokes identified by the pruning algorithm

The ITN network is particularly difficult to prune when the only criteria for creating the strokes is the angle between adjacent segments and the strokes depicted in Figure 6.22 do not give the full picture. This is because ITN is a topological network designed to aid routing and

consequently its representation of junctions can be complex and can disrupt the generation of strokes. For example, the relatively simple crossroad in Figure 6.23a is represented in ITN as Figure 6.23b. A less complex representation of the road network such as Meridian 2 (Ordnance Survey, 2014a) would be easier to prune (Figure 6.23c) but Meridian 2 has a nominal scale of 1:50K and some minor roads are omitted. These minor roads, however, may be necessary to provide context for accident hot-spots (Figure 1.5) and *are* represented in ITN.

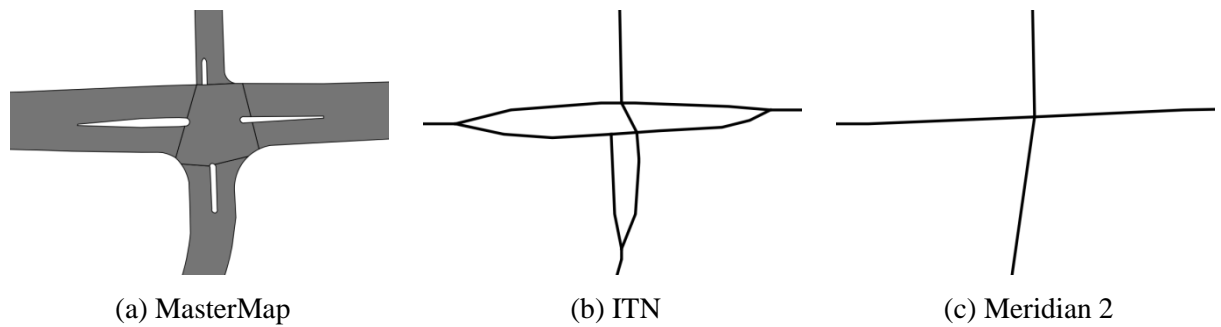


Figure 6.23 Three representations of a crossroad

Traffic islands (Figure 6.24a) tend to introduce large angles from one segment to the next thus breaking the stroke (Thom, 2005) and creating a series of single segments that are not in strokes (Figure 6.24b). The simple traffic islands in the sample network were identified (where two links share the same two nodes and are below a certain length) and removed (Figure 6.24c) and the result was that longer roads were preserved as a single stroke (For example, Figure 6.24d, stroke with an ID of 1857). However, the more complex traffic island structures (such as that shown in Figure 6.23b) were not removed due to the additional complex programming that would be required.

$$\text{lengthOfNetworkToRetain} = \left(1 - \frac{\text{DegreeOfGeneralisation}}{10}\right) \times \text{originalLengthOfNetwork}$$

Equation 6.4

So far the discussion has been restricted to pruning the road network in isolation and the first version of the pruning algorithm did just that (Table 6.5). However, for the use case it is necessary to prune the network with respect to the accident data since the road network provides context for the accident features. One option would be to retain every road segment that has an accident associated with it. The number of segments retained would depend on the time span over which the accidents are mapped; the longer the time span the more likely a segment is to have an accident on it. At relatively large scales this approach would not represent a problem but at smaller scales with a relatively long time span then the number of road segments retained would be too high for legibility. However, given the distribution and nature of the two datasets, as scale decreases, the accident data will require generalisation before the road network. This generalisation will reduce the number of features representing accidents; either directly by *selection by attribute* (section 6.5.1), or indirectly by *amalgamation* (section 6.5.2) or *aggregation* (section 6.5.3). This might relieve the problem of representing too many road features.

Input parameters		
Name	Data type	Description
MappedFeatureCollection	SimpleFeatureCollection	Mapped (road) features
DegreeOfGeneralisation	Integer	Degree of generalisation calculated from the measure algorithm results
CRS	CoordinateReferenceSystem	CRS of mapped features
MaximumAngleForStroke	Double	Limiting angle for determining whether two adjacent road segments are part of the same stroke
Output		
	SimpleFeatureCollection	Simple feature collection containing the retained (road segment) features

Table 6.5 Parameters for the basic pruning algorithm

The revised pruning algorithm was developed on the assumption that, at the stage when it was used, the accidents would be represented as polygons and the road network as lines and thus

the relationship would be *intersects* but it could be easily expanded to operate on different geometries and therefore different relationships such as *adjacent* (Figure 5.32). When using the strokes method, single segments that do not form part of a stroke would normally be omitted. However, those that intersect an accident hot-spot can provide valuable context so they are retained.

The pruning process is described in Figure 6.25. Note that the term *accident* has been used but the process could be applied to pruning roads with respect to any point feature type as long as the ontology describes its relation to the road network. Features are added until the target length of network to retain is reached (Equation 6.4).

The order of priority when determining which road segments to retain is:

1. segments in strokes that intersect with accident clusters
2. single road segments (not in a stroke) that intersect with accident clusters
3. segments in strokes that do *not* intersect with accident clusters, longest strokes first.

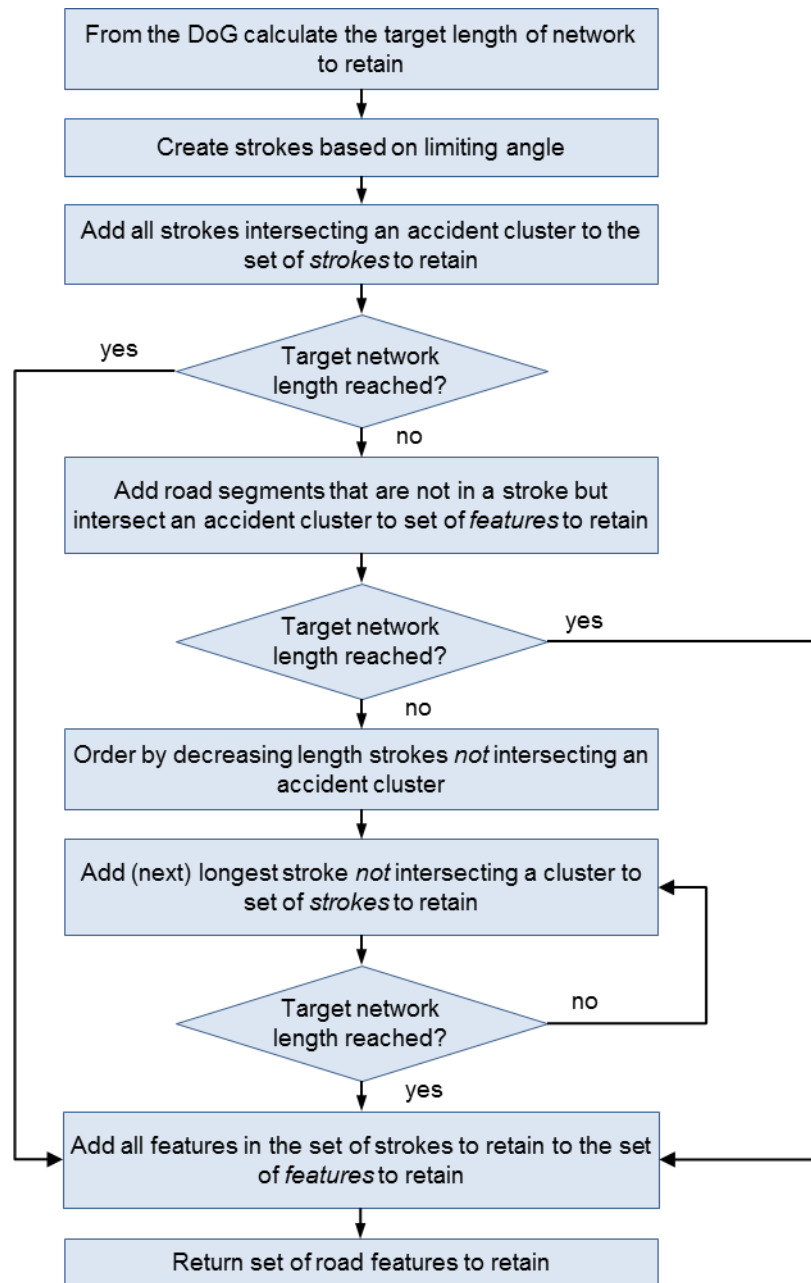


Figure 6.25 Algorithm for pruning the road network

The parameters for the enhanced pruning algorithm are described in Table 6.6. The relation to respect is extracted from the ontology, before the algorithm is called, based on the respective geometries and feature types of the two feature collections. The Manchester OWL syntax to return the relation (based on Figure 5.34) is

```

SpatialRelation and hasThematicFeatureType some AccidentFeatureType and
hasThematicGeometry some AreaGeometry and hasSupportFeatureType some
RoadFeatureType and hasSupportGeometry some AreaGeometry

```

This should return the class *AccidentsIntersectsRoad*. The next step is to identify the parent of the class, that is *intersects* (which is not specific to roads and accidents). This is the relation

passed to the algorithm (Table 6.6). The *IntersectMeasureAlgorithm* defined in the ontology (Figure 5.34) is implemented by the *intersects* method of JTS Topology Suite¹³ *geometry* class. Currently it is the responsibility of the pruning algorithm to determine the functionality that will implement the *intersects* test. It might be preferable for the mapping engine to determine this functionality and pass it to the algorithm.

Input parameters		
Name	Data type	Description
MappedFeatureCollection	SimpleFeatureCollection	Mapped (road) features
DegreeOfGeneralisation	Integer	DoG calculated from the measure algorithm results
CRS	CoordinateReferenceSystem	CRS of mapped features
MaximumAngleForStroke	Double	Limiting angle for determining whether two connected road segments are part of the same stroke
RelationToRespect	String	Name of the relationship to be respected e.g. "intersects"
ThematicFeatureCollection	SimpleFeatureCollection	Features that have to be "respected" e.g. accident clusters as polygons
Output		
	SimpleFeatureCollection	Simple feature collection containing the retained (road segment) features

Table 6.6 Parameters for the pruning with respect to accidents algorithm

The effect of pruning the road network while respecting the relation with the accidents can be seen in Figure 6.26¹⁴. The single segment, *x* (Figure 6.26b) and the stroke *yy* (Figure 6.26b) that would normally be omitted when the only consideration is stroke length (Figure 6.26a) are retained because they intersect the accident cluster.

¹³ JTS is implemented by GeoTools to provide the Geometry data structure.

¹⁴ The scale depicted is not the scale at which the features were generalised. The scale has been exaggerated for improved presentation.

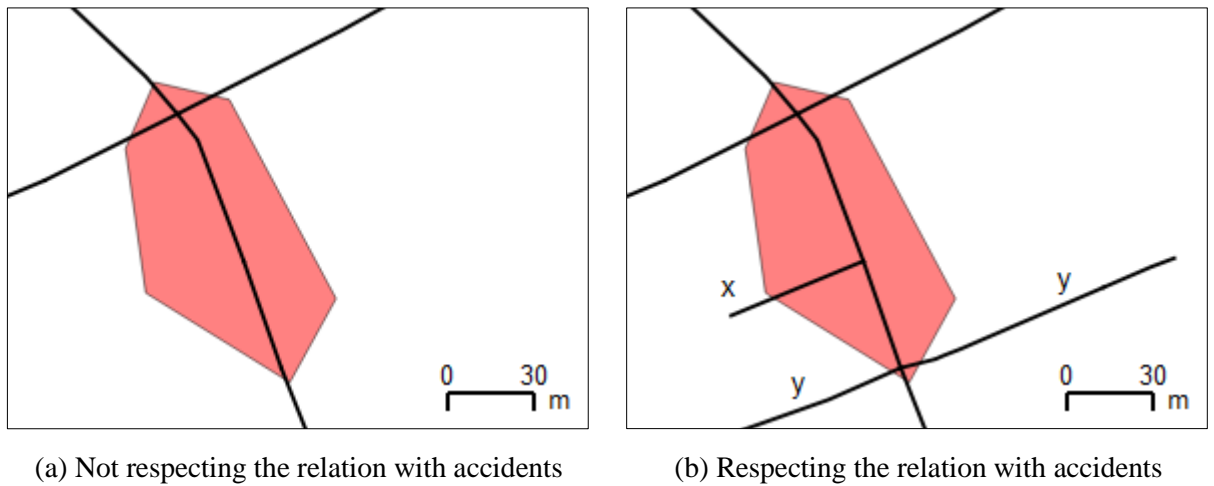


Figure 6.26 Pruning the road network

6.7 Platform design

This section describes how the ontology and the algorithms described above were utilised in a prototype on-demand mapping system. The aim was to transform the *architecture design* developed in Chapter 4 (Figure 4.17) into a *platform design* as prescribed by the CommonKADS methodology (Schreiber et al., 2000). The platform design is depicted in Figure 6.27. The on-demand mapping system was implemented by developing a Java application. In the implementation the reasoning agent described in the architecture design is included as part of the Mapping Engine and therefore is not depicted separately. Although integrated in the Java application, the measure and transformation algorithms could be implemented externally, as web services for example.

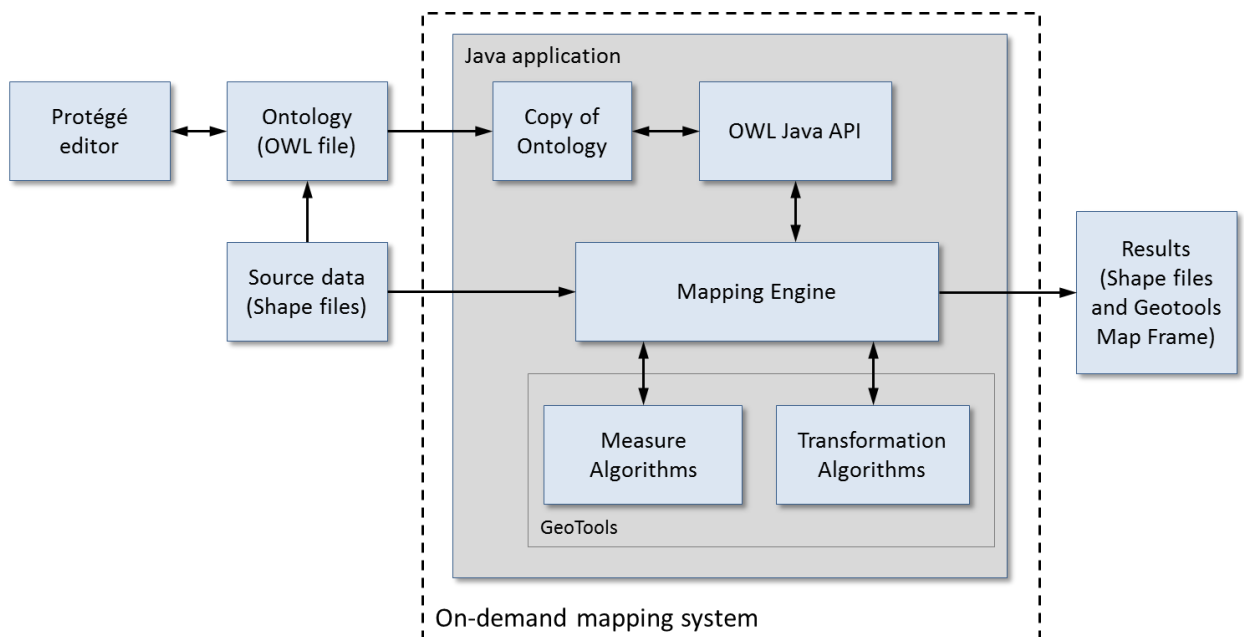


Figure 6.27 On-demand mapping system platform design

6.7.1 Interacting with the ontology - the OWL API

The only role of the Protégé editor is to create and edit the ontology; it has no part in the on-demand mapping process. Because the generalisation process makes modifications to the ontology, by creating new individuals, the on-demand mapping system takes a copy of the ontology at start-up. There is an individual in the original ontology that describes each source feature collection. However, when the accidents are amalgamated into polygons, for example, a new individual is created that describes the output feature collection that has a different set of features with a different geometry.

Interaction between the Java application and the ontology, strictly the copy of the ontology, is done via the OWL API (Horridge & Bechhofer, 2011). Most of the calls to the ontology consist of queries in the form of Manchester OWL syntax (Horridge & Patel-Schneider, 2008). The syntax is verbose but human-readable, which makes development easier. The following is an example of query string generated by the Mapping Engine and executed against the ontology, which will return a list of measure algorithms meeting the specified criteria:

```
MeasureAlgorithm and measures some HighFeatureDensity and hasInputGeometry  
some AreaGeometry
```

More specifically, the API can be instructed to return either classes or individuals that meet the criteria. Throughout this chapter, the OWL objects discussed are *classes* unless stated otherwise.

6.7.2 Implementing the Problem Solving Methods

The OWL queries are based on the *inference tasks* defined in the *inference layer* (Figure 4.16). Some of the tasks were broken down into multiple queries, with the output of one query being used as an input to another. For example, the task of identifying an appropriate measure algorithm involved three queries to match each inference action in the problem solving method (Figure 4.9).

6.7.3 Workflow – control knowledge

The problem solving methods, now expressed as OWL queries have to be sequenced in order to create a map of the accidents and roads at the user's selected scale. This is the control knowledge that describes how the components are utilised each time the user changes scale.

The control knowledge is built in to the mapping engine and not represented separately.

Figure 6.28 lists the initial steps for the mapping engine.

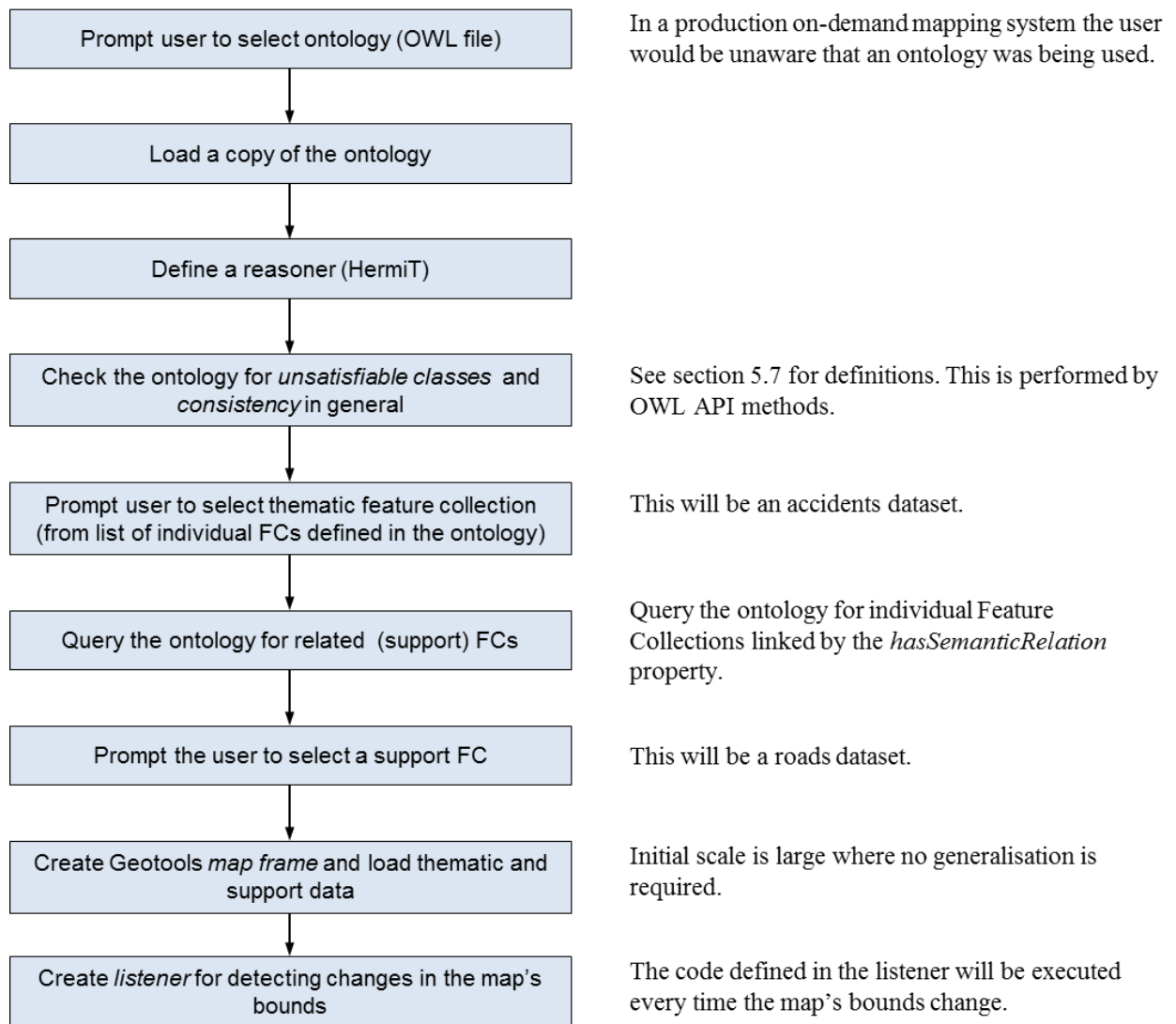


Figure 6.28 Preliminary steps for the mapping engine

At the conclusion of the preliminary steps the user is presented with a map of the accidents and roads at a large scale, centred on the accidents dataset (Figure 6.29). If the user uses any of the navigation tools – zoom in, zoom out, display full extent and pan – the code defined in the map bounds change event listener is executed. The steps executed in the Mapping Engine are listed in Figure 6.30. For the sake of simplicity the flowchart shows only one possible condition being processed and only one possible symptom for that condition but the software can manage multiple conditions and symptoms for the same feature collection.

The current scale is determined since it is required as an input parameter to the measure algorithms and most of the transformation algorithms. The rest of the workflow is repeated for each feature collection mapped; the accident features, as the thematic features are processed first and then the road features. Generalisation is incremental in that once a feature collection has been processed a geometric measure is applied to the *generalised* features and not the *original* mapped feature collection. This is repeated until there are no geometric conditions in the feature collection. The same measure algorithm is not necessarily used since the nature of the features mapped may have changed or new conditions may have been introduced by the previous generalisation step. The application of a transformation algorithm will change the data (e.g. from a cluster of point features to an area feature). However, the changes enacted by the transformation have to be reflected in the ontology; this is why a working copy of the ontology is made, to protect the original (Figure 6.27). The semantics of the features may have changed as well as the geometry. To be precise, the feature type of a set of amalgamated accidents is no longer *AccidentFeatureType*. The changes in the semantics may well effect what measure and transformation algorithms are applicable in subsequent iterations (if required). This process is known as *semantic propagation* (Janowicz et al., 2010) and is not yet fully managed in the prototype.

The entire process is complete when there are no geometric conditions identified by the measure algorithms for either of the feature collections. If the user changes scale again then the process is repeated, starting with two new mapped feature collections.

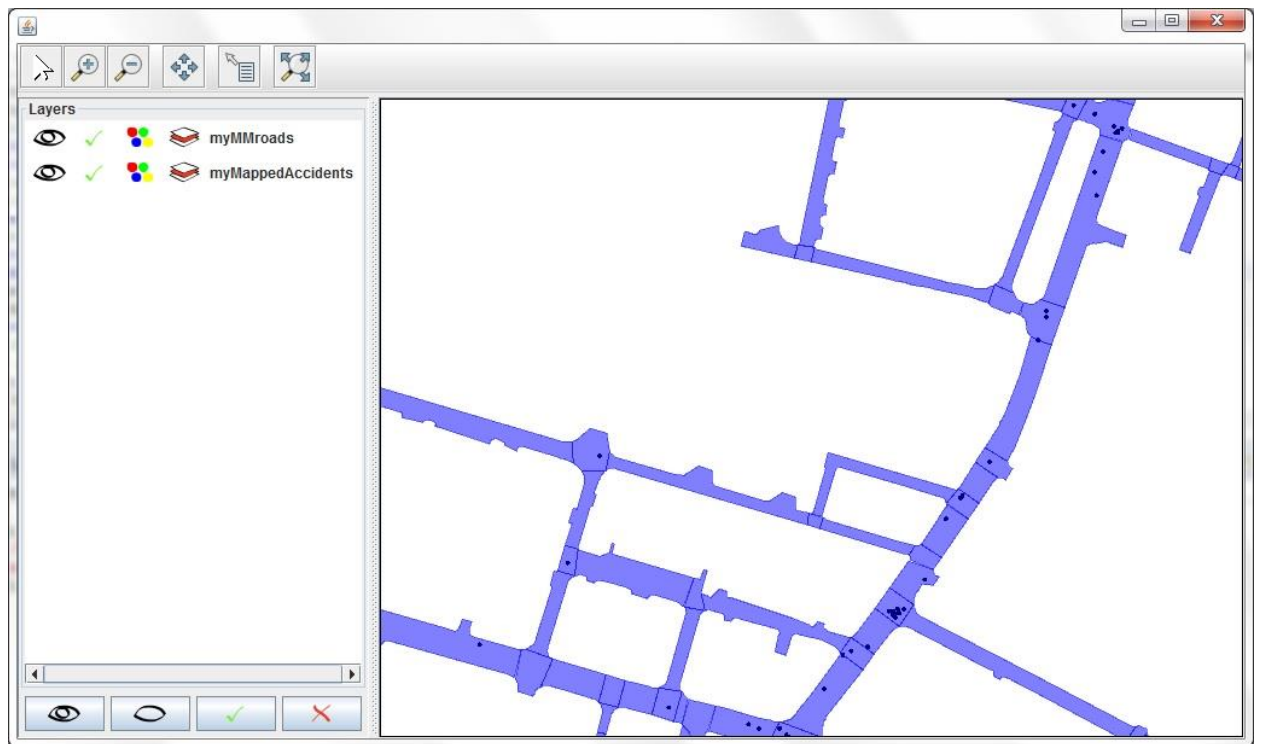


Figure 6.29 Initial view of the on-demand mapping system at scale 5.0 pixels/m (the dots represent accidents)

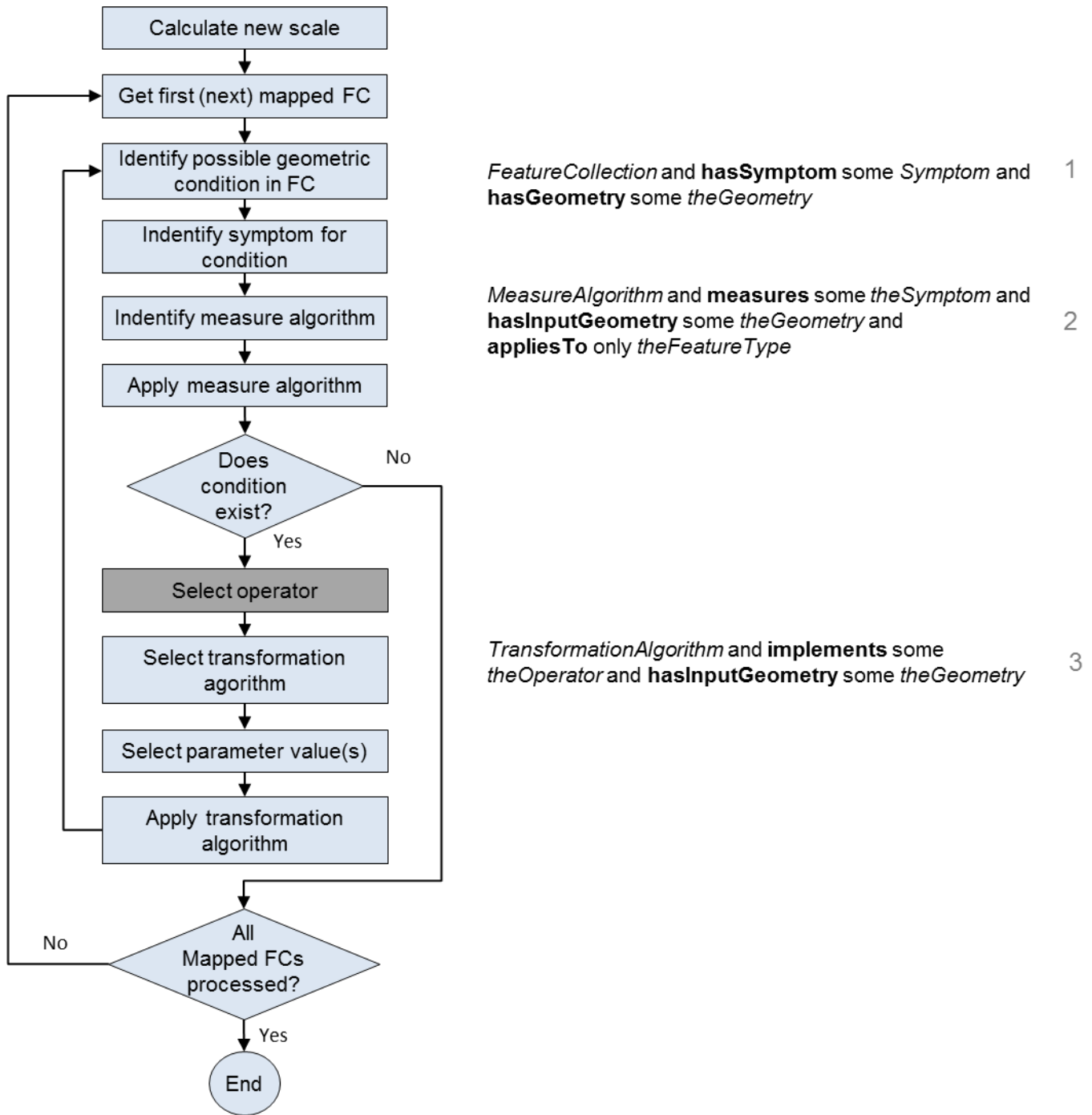


Figure 6.30 Simplified steps in the *map bounds changed* event listener (FC = Feature Collection)

The following are some observations about the OWL queries (the numbering follows that in Figure 6.30 where classes are in *italic* and relationships in **bold**):

1. This will return any Feature Collection class that has any symptom and the geometry of the Mapped FC. For example, the *CongestedFeatureCollection* class will have the symptom *HighFeatureDensity*. This step is repeated following application of the transformation algorithm because new a new condition may have been introduced by the generalisation.

2. This query will return the MeasureAlgorithm subclasses that measure the specified symptom (extracted in the previous step), and have the specified geometry (which is that of the Mapped FC). The final clause restricts the results to specialist measure algorithms as discussed in section 5.7).
3. This query returns any algorithm class that implements the specified operator and has the specified input geometry. Specifying an operator alone is not sufficient since different algorithms will generalise different geometries. Once an algorithm *class* has been identified, the Java function that is represented by an *individual* in that class will be called.

The *select operator* step, highlighted in Figure 6.30, is described in detail in Figure 6.31. This process is handled by a separate function (Table 6.7). The aim of this function is to return a list of candidate operators that can remedy a given symptom in a given feature collection. The process implements the concepts of an operator *requiring* a particular type of feature collection (Figure 5.18) and a feature collection *forbidding* a particular operator (Figure 5.19 and Figure 5.22). The process as described may seem too complex and could be simplified by refining the OWL queries (listed to the right of the flowchart).

Input parameters		
Name	Data type	Description
FeatureCollection	SimpleFeatureCollection	Feature Collection to generalise (represented by an OWL individual)
Symptom	OWL Class	Symptom identified by measure algorithm e.g. <i>HighFeatureDensity</i>
Geometry	String	Geometry of FeatureCollection
CRS	CoordinateReferenceSystem	CRS of FeatureCollection
ParentClass	OWL Class	Parent class of FeatureCollection e.g. <i>AccidentFeatureCollection</i>
Output		
List of operators	Set of OWL Class	Operators that may remedy the symptom

Table 6.7 Parameters for function to identify candidate operators for a particular symptom

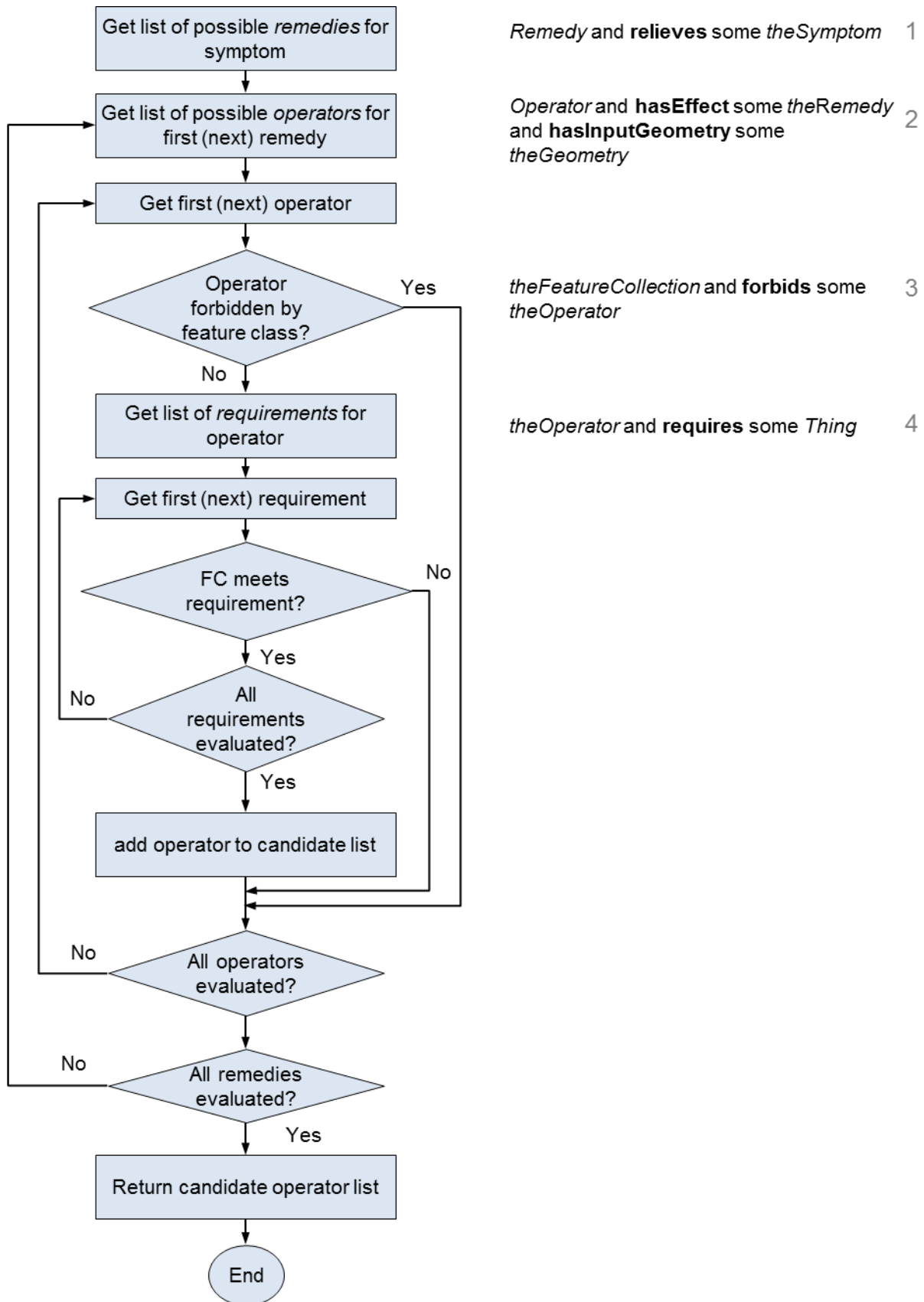


Figure 6.31 Flowchart for returning list of appropriate operators for the given feature collection (FC) and symptom

The following are some observations about the OWL queries (the numbering follows that in Figure 6.31):

1. *theSymptom* is the Symptom class specified by the input parameter (Table 6.7).
2. *theGeometry* is the Geometry class specified by the input parameter (Table 6.7).
theRemedy is the current *Remedy* derived from query 1; *FeatureCountReduction*, for example.
3. *theFeatureCollection* is the parent class specified by the input parameter (Table 6.7).
theOperator is the current Operator class.
4. All classes in the ontology are members of the *Thing* class. It is used here since the object of the *requires* property can be one of a number of classes. The query will discover the requirement but inference is used to check whether the current Feature Collection (represented as an individual) is in the required class.

If the ontology suggests more than one operator to resolve the condition then the user is presented with a drop-down box (Figure 6.32). Currently there is a one-to-one matching of operator to algorithm.

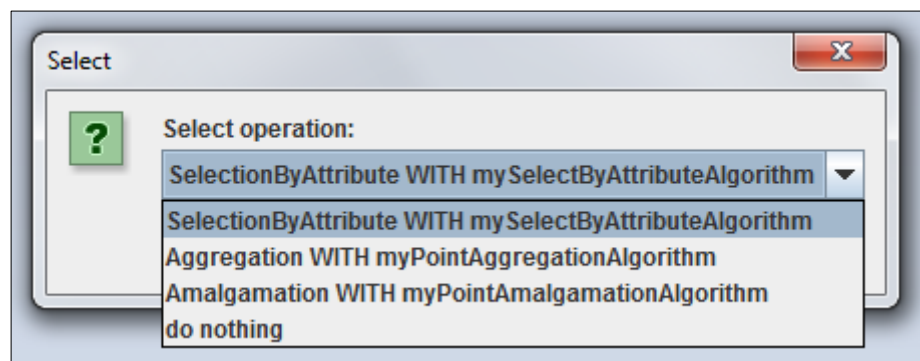


Figure 6.32 User selection of an operator/algorithm

6.8 Results

Results will be described for maps at three scales, for the three different types of user identified in the use case (Table 5.1); a relatively large scale that displays the details of a junction (approximately 1:500 scale), an intermediate, neighbourhood scale (approximately 1:5K), and a relatively small scale that extends to a city centre (approximately 1:20K). The results were saved as Shape files and imported into QGIS for clearer visualisation than can be provided by the GeoTools map frame environment (Figure 6.29).

6.8.1 Large scale – Highways engineer

At a relatively large scale (8.0 pixels/m, approximately equivalent to 1:500) the detail of road junctions can be seen (Figure 6.33, p149). The point feature density measure algorithm (*findHighPointDensityClusters* described in section 6.3.1) and the road area density measure

(*findHighDensityAreaCrossroadClusters* described in 6.3.2) were automatically selected and applied and no conditions were identified in the mapped FC.

At this scale it is possible to see that the accidents in the central T-junction are mostly at the south of the junction. It can also be seen that this junction has a relatively high number of accidents in comparison to the other two T-junctions mapped. Chapter 7 will discuss how to provide more context to determine why this might be the case.

6.8.2 Medium scale – Parent

After the user (the parent) selects the accident dataset the system suggests two road datasets (the ITN dataset and the MasterMap dataset) as support feature collections since both individuals share an *isOn* relationship with the accident (Figure 5.31). In this example, at a scale of 1.0 pixels/m (approximately equivalent to 1:5K), the ITN road network was selected as a base feature set (Figure 6.34). This led to the *findHighDensityLineCrossroadClusters* being selected by the system which found that there was no need to generalise the road features. However, a number of congested accident clusters were identified by the measure algorithm (Figure 6.34). The ontology offered one of the three point feature generalisation algorithms (section 6.5) and aggregation was selected by the user (Figure 6.35). A DoG of 6 was derived by the system based on the number of problem features identified but this removed too many features for the user's preference and a DoG of 3 was used.

It can be seen immediately that when walking from *A* to *B* the route via junction *X* should be safer than that via junction *Y* (Figure 6.35). The map therefore serves the aim of a parent finding a safe route for a child to walk to school.

At a similar scale the *selection by attribute* algorithm (section 6.5.1) was applied but at the suggested DoG of 4 (derived from the problem features highlighted in Figure 6.36) all of the mapped features were retained and only by applying a DoG of 9 could the feature count be reduced (Figure 6.37). The reason for this can be seen in Figure 6.9. This highlights the reliance of this operator on having an even distribution of values in the importance attribute.

6.8.3 Small scale – Road safety expert

At this scale, 0.15 pixels/m (approximately equivalent to 1:20K), the full extent of the test data is shown. The problem accident features can be seen in Figure 6.38. The roads were initially represented as polygons in the MasterMap dataset and at this scale the system identifies problem road features (Figure 6.39) using the *findHighDensityAreaCrossroadClusters* algorithm (section 6.3.2). This topographic dataset is not defined as a *Network* in the ontology therefore pruning is not suggested at this stage. In

fact, the only operator suggested is *collapse*. This operator is defined in the ontology as reducing feature density by reducing feature *size* rather than feature *count* as with pruning. Collapse was applied (section 6.6.1) and since the output is line features the system applied the *findHighDensityLineCrossroadClusters* algorithm (section 6.3.2) and problem features were highlighted (Figure 6.40). Since the ITN network represents a network and consists of line data the system suggested a *pruning* algorithm.

The first set of results can be seen in Figure 6.41. Both datasets were generalised with the DoG derived from the respective measure algorithms. The amalgamation algorithm was applied to the accident features with a DoG of 9. It can be seen that the accidents were over-generalised and only the numerically largest clusters were retained (section 6.5.2). The road network was pruned with the suggested DoG of 4 and it can be seen that the network is not fully connected but the aim was not to develop the perfect pruning algorithm. What is important is that any road segment intersecting an accident cluster polygon was retained. In a normal pruning algorithm “dangling” links will be removed but here they provide context for the accident data.

The retention of such road segments can, unfortunately, cause areas of high density road features to be retained and thus identified as problem features when the measure algorithm is run again, which means that the road network can never be generalised to satisfaction. This can be resolved in a number of ways; firstly by simply accepting the initial level of pruning. The system allows for this by giving the user of “doing nothing” when a feature set is found to have a condition. Another option would be to breaking the relation between roads and accidents and remove road segments that intersect an accident polygon. Finally, a higher degree of generalisation to the accidents than is suggested could be applied and thus reducing the number of accident polygons mapped.

The process was repeated (at the same scale) but with the accidents amalgamated with a DoG of 5 (Figure 6.42). This gave a better picture of the accident hot-spots. In the final set of results, the DoG of 5 for the accidents was retained but used as an input to aggregation (Figure 6.43). The roads were pruned with a DoG of 7 and not the suggested value of 4. It could be argued that this provides the most useful results. If the user zooms in to the bounds of the numerically largest cluster (highlighted with a dashed line in Figure 6.43) then the user gains more detail (Figure 6.44). At this stage some labelling with road names would be useful.

It could be argued that the “small” scale here is not that small and it is unfortunate that the limitations of the algorithms developed prevents the system from being tested at a scale where larger extents could be mapped

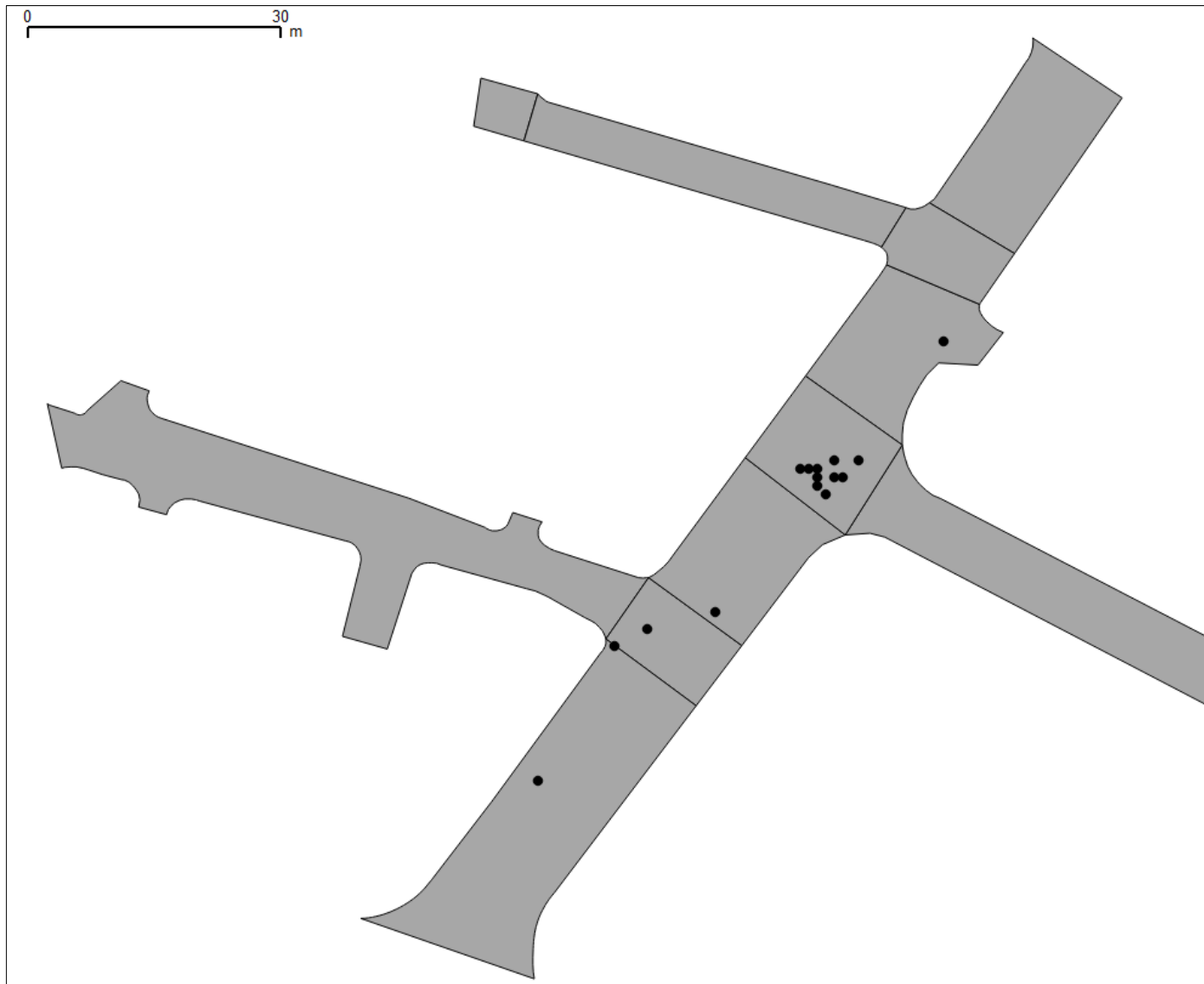


Figure 6.33 Scale of 8.0 pixels/m no problem accident or road features identified

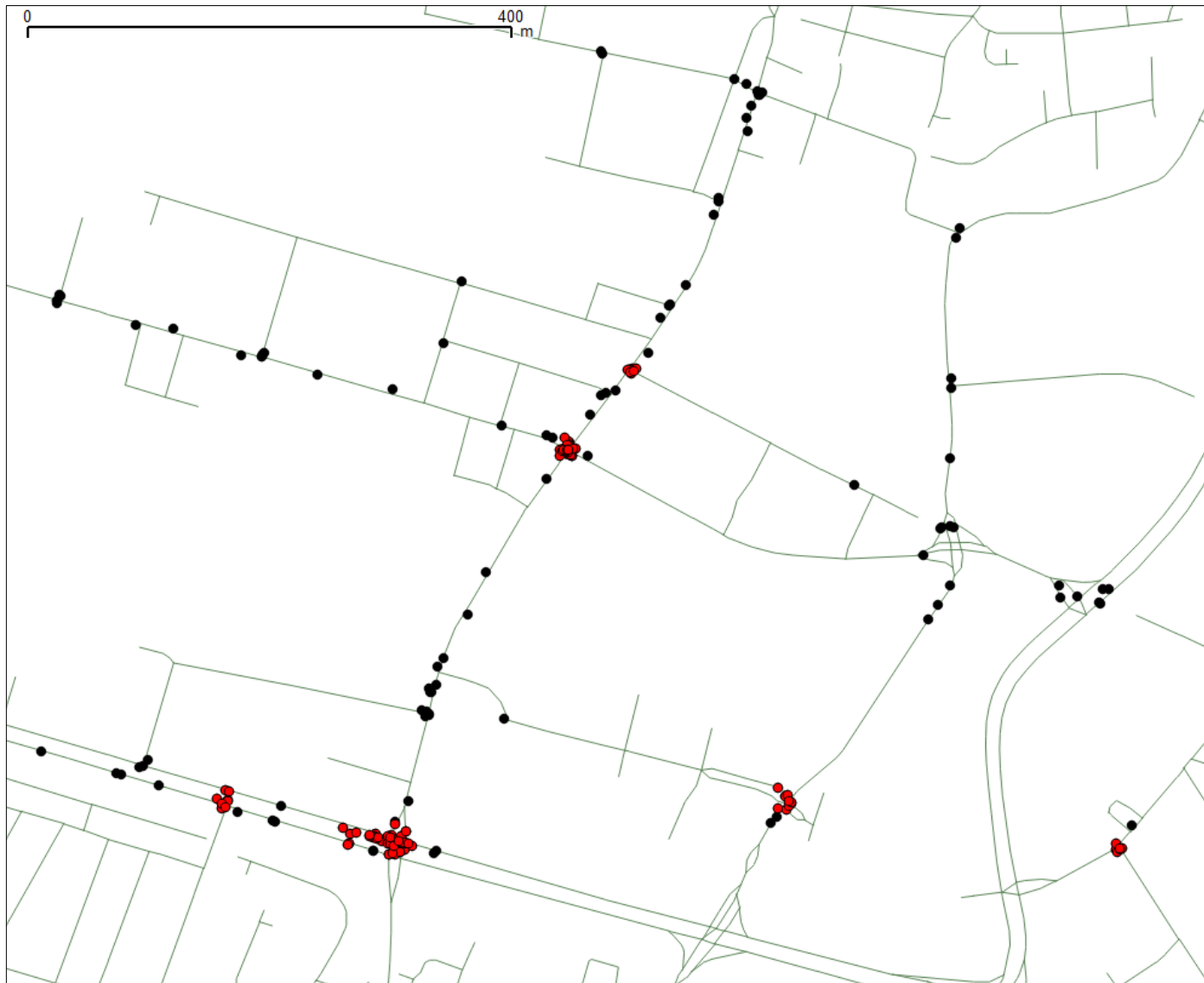


Figure 6.34 Scale 1.0 pixels/m. Problem accidents highlighted in red. (road line features dataset chosen at start)



Figure 6.35 Scale 1.0 pixels/m. Accidents aggregated with a DoG of 3 (suggested values of 6). No problem road features identified



Figure 6.36 Scale 0.94 pixels/m Problem accident features highlighted in red

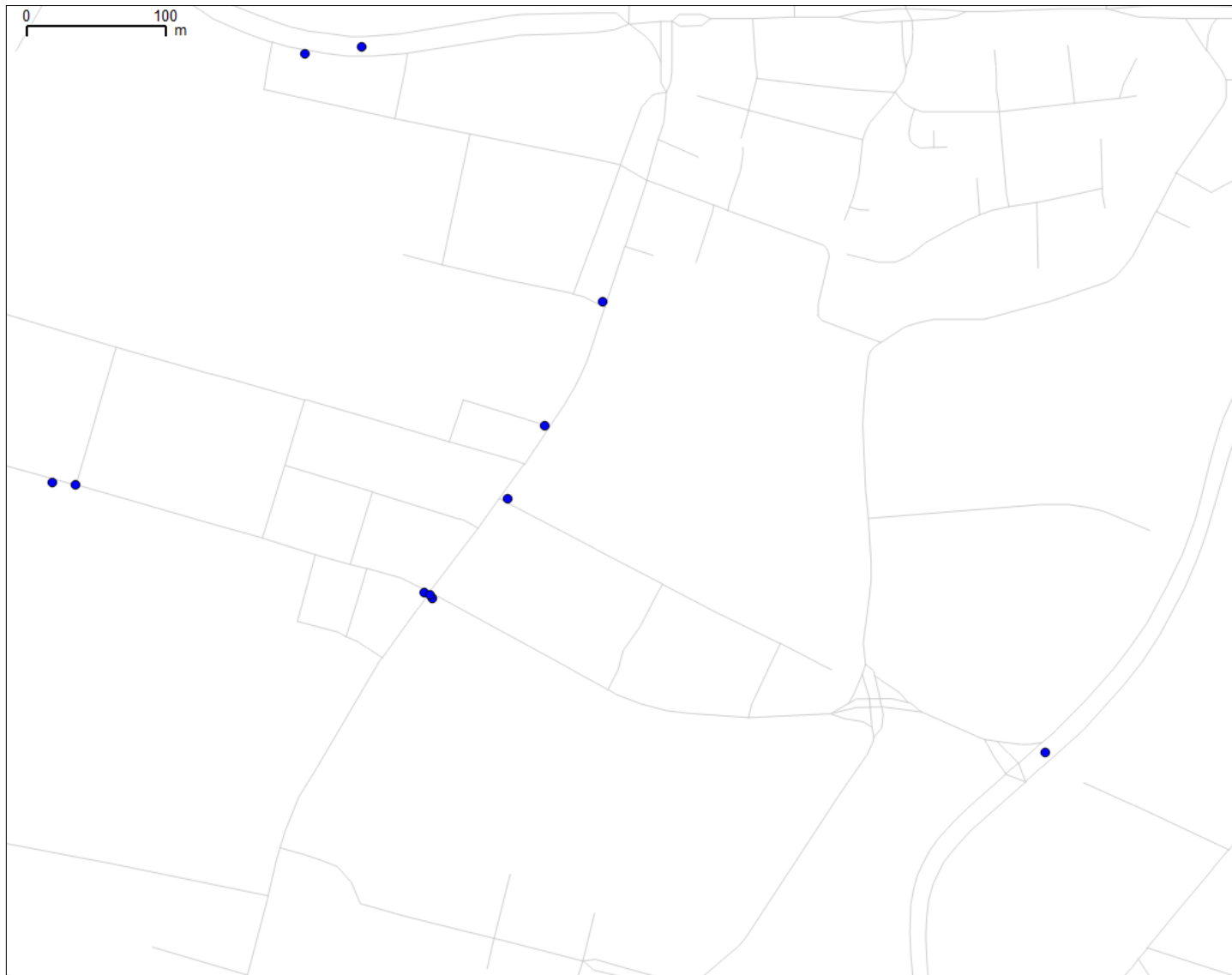


Figure 6.37 Scale 0.94 pixels/m. Selection By Attribute applied to accidents with a DoG of 9 (suggested value was 4)

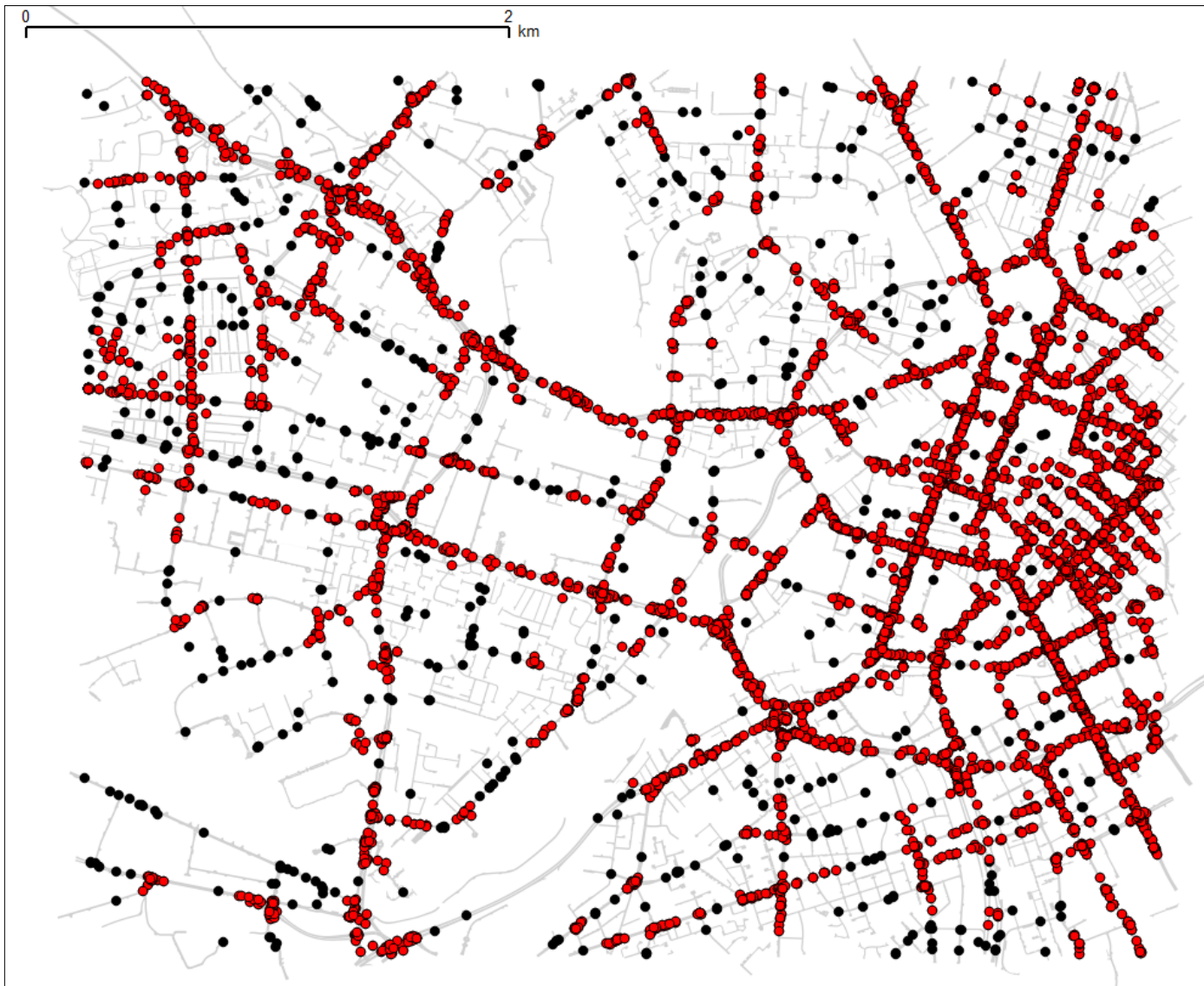


Figure 6.38 Scale 0.15 pixels/m. Problem accident features highlighted in red



Figure 6.39 Scale 0.15 pixels/m. Problem road area features (MasterMap) highlighted in red



Figure 6.40 Scale 0.15 pixels/m. Problem road line features (ITN) after collapse highlighted in red



Figure 6.41 Scale 0.15 pixels/m. Accidents amalgamated with a DoG of 9 (suggested value was 9). Roads pruned with DoG of 4 (suggested value was 4)

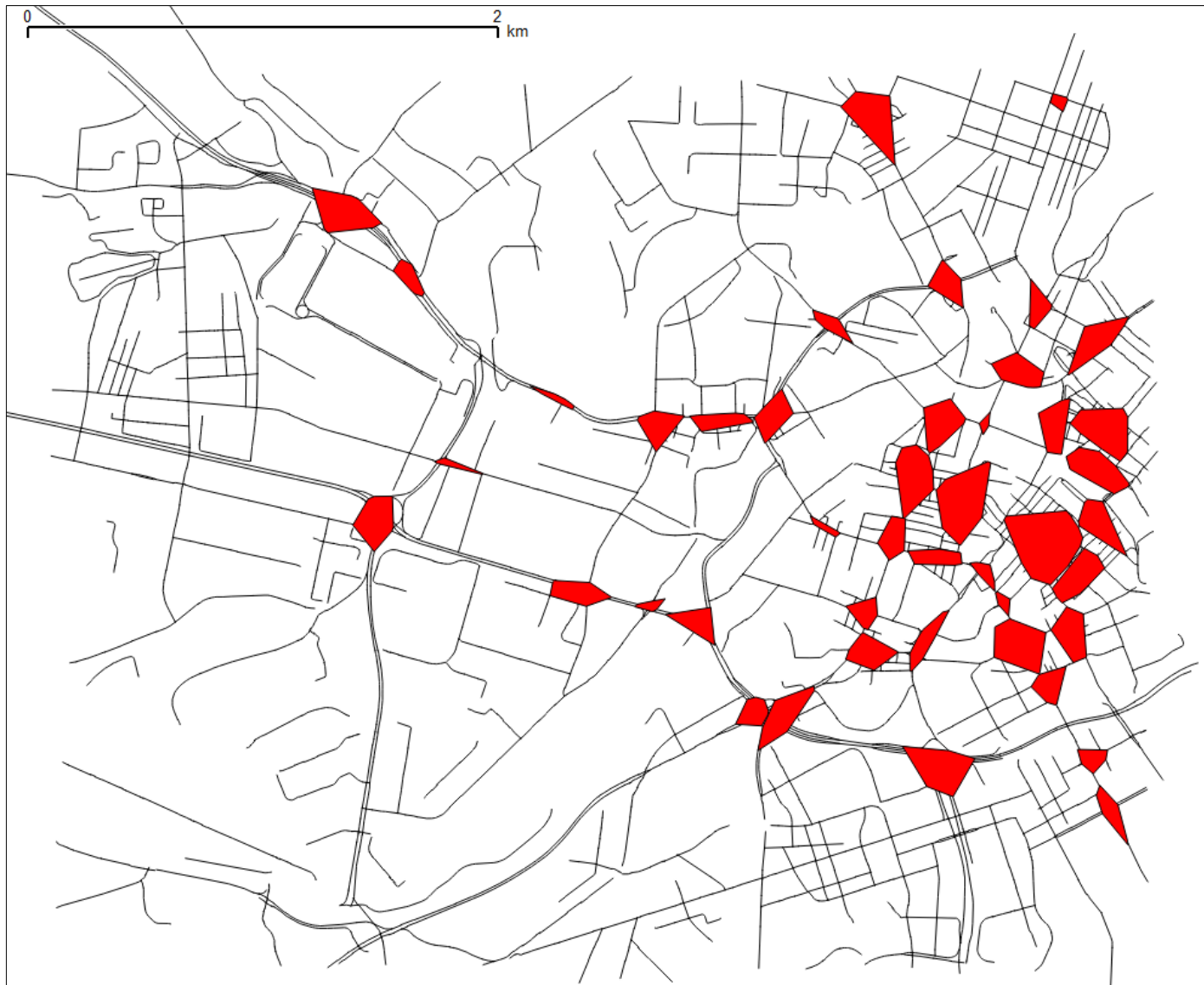


Figure 6.42 Scale 0.15 pixels/m. Accidents amalgamated with a DoG of 5 (suggested value was 9). Roads pruned with DoG of 4 (suggested value was 4)

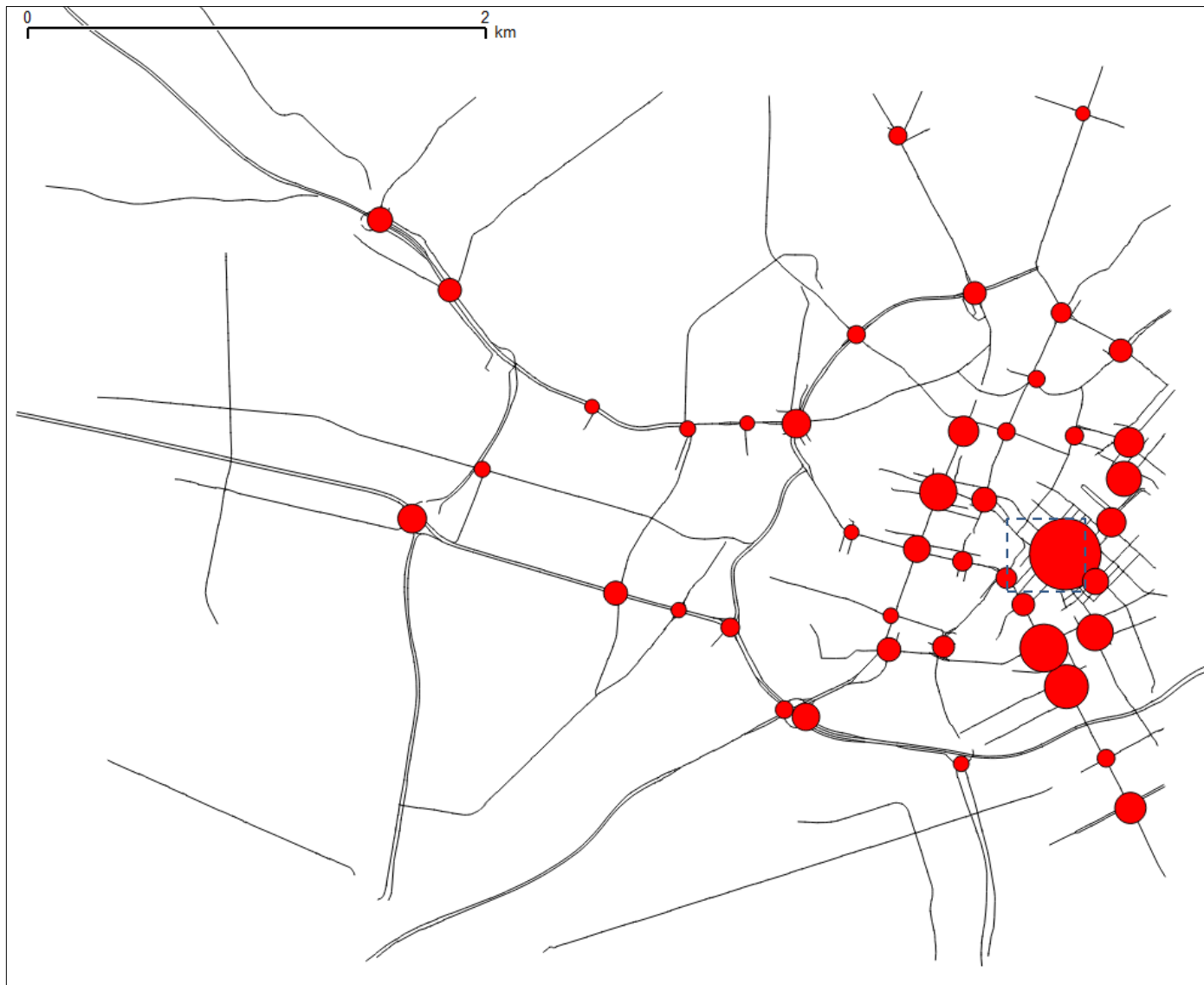


Figure 6.43 Scale 0.15 pixels/m. Accidents aggregated with a DoG of 5 (suggested value was 9). Roads pruned with DoG of 7 (suggested value was 4)

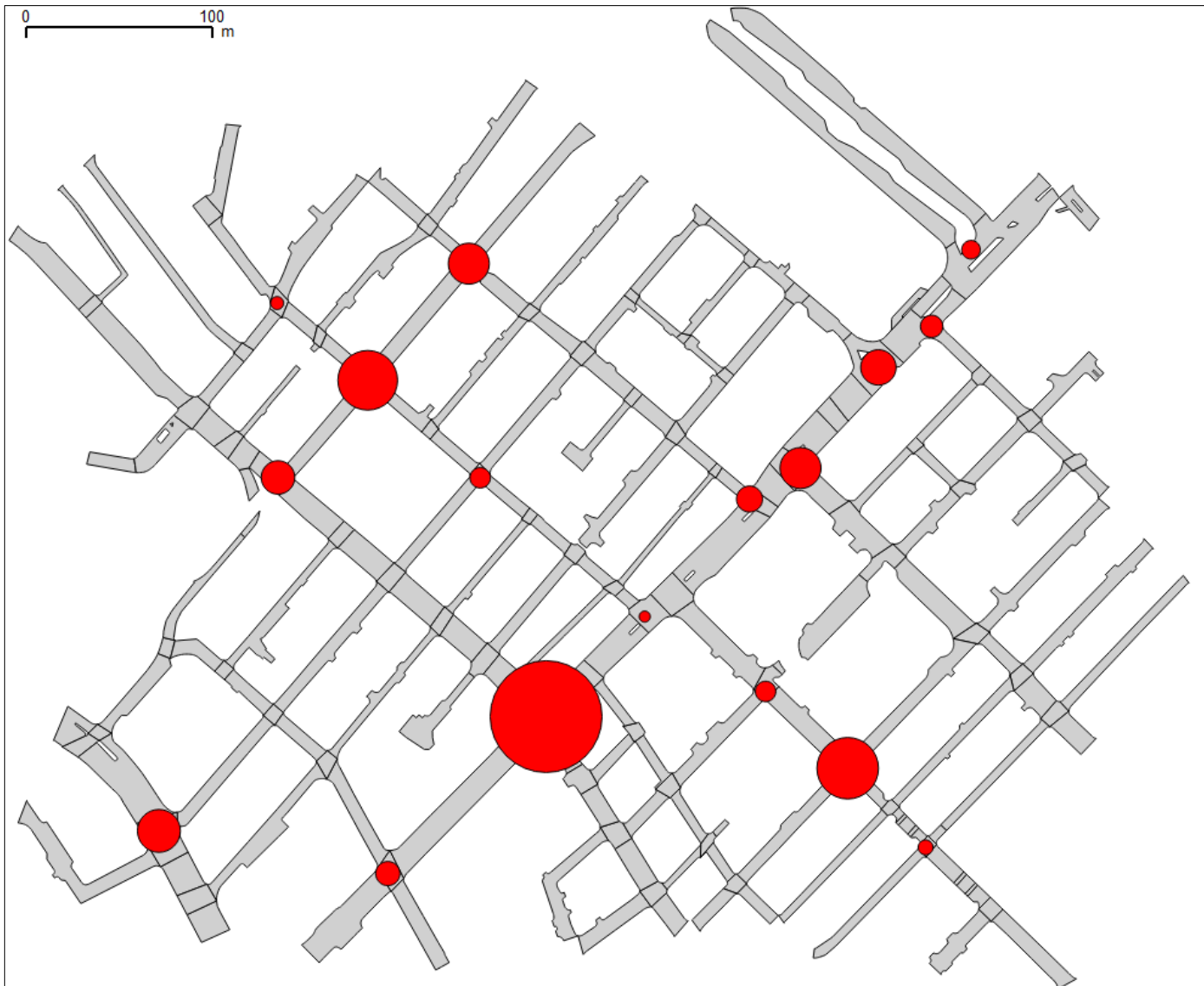


Figure 6.44 Scale 1.69 pixels/m. Accidents aggregated with a DoG of 5 (suggested value of 5)

6.9 Conclusions

Chapter 7 will discuss the results in general but this section will assess the methodology and evaluate the algorithms that were developed.

6.9.1 Methodology assessment

To what extent was the application of the *CommonKADS methodology*, as described in Chapter 4, successful? The CommonKADS methodology was specifically designed for knowledge-based systems (KBS) and the on-demand mapping system is only partially a KBS, mixing inference and non-inference tasks. This led to a less than perfect mapping of the *architecture design* (Figure 4.17) to the *platform design* (Figure 6.27).

The main advantage of the commonKADS methodology was how it aided the development of the prototype by allowing the organisation and definition of the control knowledge and making a distinction between *inference* knowledge and *task* knowledge (Figure 4.3). The inference knowledge represents the demands on the ontology and the task knowledge represents the expression of the McMaster and Shea generalisation model. Whereas the *domain* knowledge was held by the ontology, the *control* knowledge was embedded in the Java code; the inference knowledge, explicitly, as a loose collection of Manchester syntax OWL queries and the task knowledge, implicitly, within the control flow of the prototype software. This representation contradicts the desire to make as much of the generalisation knowledge as possible shareable.

Further consideration of how the different components interact with the ontology is required. Whether, for example, the pruning algorithm should interact directly with the ontology. Currently the *intersects* test is implemented in the algorithm using the GeoTools Java library. The test, however, could be exposed as a web service. The algorithm could then interact with the ontology to locate a service that implemented the test or the mapping engine could locate the service and pass the address to the algorithm.

The development of a prototype was defined as necessary part of the evaluation phase of the *ontology design methodology* (section 3.7.3). Although no new top-level concepts were added during the development of the prototype some problems with the ontology design were identified in a number of places during the development of the prototype, which led to new subclasses and relationships being added to the ontology. In particular the definition of semantic and spatial relationships between the two feature types was initially too superficial to meet the needs of the prototype and had to be made more sophisticated. This is an example

of the feedback loop described in the ontology design methodology (Figure 3.6). A proper record of the changes made to the ontology specifically during the prototype development phase was not kept, but there were enough changes to conclude that the final ontology would look very different if the prototype had not been attempted. It can be concluded that the development played a vital role in the evaluation of the ontology.

6.9.2 Evaluation of the measures and the Degree of Generalisation concept

The measure algorithms identified the *problem*, congested features, and the Mapping Engine calculated the DegreeOfGeneralisation as a linear function of the number of problem features. When used with the aggregation and amalgamation algorithms the application of the DoG removes too many accident features (compare Figure 6.41 and Figure 6.42). It could also be concluded that the application of the suggested DoG removes too few road features (Figure 6.41).

As it stands the application of the measures and the derivation of the DoG lacks sophistication. The *total* number of problem features was used to derive the DoG and the number of congested clusters and their distribution was not considered. As Stigmar and Harrie (2011) concluded, the application of a single measure to determine legibility is not sufficient and a combination of measures is necessary. Also the measures were applied to each mapped feature collection separately and it might be beneficial to consider holistic measures.

The measure algorithms identified the problem features but the generalisation algorithms were applied to the complete *mapped* feature collection. So, for example, accidents that were not identified as problem features were still removed. This is not necessarily a disadvantage. If we consider that the aim of the generalisation of the accidents is to identify accident hot-spots then the removal of non-problem features is desirable.

Despite these limitations, the application of measures combined with the DoG concept leads to generalisation based on *measured* need. This is in contrast to the *Radical Law* (Töpfer & Pillewizer, 1966) and its variation, the *Simplification ratio* (Foerster et al., 2007b), which are based on *expected* need, where the amount of generalisation is simply based on the number of mapped features and the ratio of the source and target scales (section 4.4.2). Therefore, it can be concluded that, despite the limitations in the current implementation, the DoG concepts merits further investigation.

6.9.3 Evaluation of the transformation algorithms

Improvements could be made to the transformation algorithms, in particular the pruning algorithm. For example, disconnected strokes could either be connected to the rest of the network by using a shortest path measure or removed if they do not interact with clusters. The strokes were created by considering only the angle between adjacent segments. Other factors such as road name and road class could be used when judging whether two adjacent segments were part of the same stroke. For example, adjacent road links that are in the same road class could be allowed to have a higher limiting angle than those in different classes (Thomson & Richardson, 1999). A mesh approach (Chen et al., 2009) or a combined stroke and mesh approach could also be investigated, particularly the multiple-scale method of Li and Zhou (2012).

The pruning algorithm (Figure 6.25) retains *all* strokes and single road segments that intersect an accident cluster. This assumes that the combined length of these features does not exceed the target length of the network as determined by the DegreeOfGeneralisation (Equation 6.4). There were instances, especially at smaller scales, where the target length was exceeded and the commitment to respect the relationship with the accidents conflicted with the need to respect the DoG required in the road network (Figure 6.45).

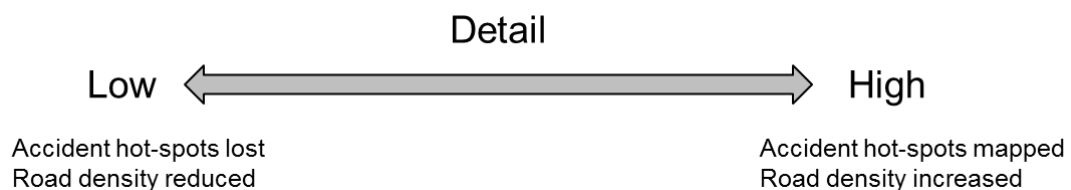


Figure 6.45 Compromise between showing all of the accident clusters and reducing road density

It might be useful to capture the balance between low and high detail in the ontology as a part of a concept of user requirements to determine if the user was interested in detail or in an overview. The concept of user requirements will also help with the problem of deciding which of the generalisation operators is appropriate for the accident features. This will be discussed in the following chapter. Although the pruning algorithm is not without weaknesses it represents a rare attempt to prune a road network while respecting its relation with another feature type.

Despite the limitations discussed above, the system fulfilled the aim of providing junction detail (although the mapping of more contextual features, such as traffic lights would be

useful), identifying safe route to schools and identifying accident hot-spots at city centre scale.

The aim of this research was to fully automate the generalisation process. But this aim ignored subjectivity in mapping. This is evident when questions such as “What is congestion?” and “How much generalisation is required?” are considered. The first question can be resolved by allowing the user to adjust the *featureWeight* factor, used to determine the clusters, and the second by allowing the user to adjust the *Degree of Generalisation* applied. However, would we expect the selection of generalisation algorithms to be completely automated? Generalisation of the road network was relatively straightforward; operators proposed were limited to *collapse* and *pruning*. However, the user was given a choice of three operators to generalise the accident data. Allowing the user to influence the mapped output is one of the benefits of on-demand mapping and the next chapter will consider ways of, not limiting that choice, but embracing it.

7 An evaluation of the ontological approach

7.1 Introduction

The creation of the ontology (Chapter 5) and its implementation as part of an on-demand mapping system (Chapter 6) raised a number of issues that require further consideration (section 7.2). Although the ontological approach proved successful for the road accident use case it is important to assess how the approach could be applied to other use cases (7.3).

7.2 Outstanding issues

7.2.1 Knowledge representation

Existing automatic generalisation systems such as those that implement agent-based techniques were criticised earlier (section 2.5) for embedding generalisation knowledge within software and thus restricting its use. The aim of this research was to formalise generalisation knowledge in an ontology, however a lot of knowledge is held in the Mapping Engine and in the algorithms (Figure 6.27). The knowledge held in the Mapping Engine includes

- the formula used to calculate the Degree of Generalisation (Equation 4.1)
- the inference tasks (expressed as Manchester OWL queries)
- the procedural knowledge that is the control flow of the mapping engine, which is based on the Problem-Solving Methods (section 4.3.2) and the sequence dictated by the McMaster and Shea model.

The knowledge held in the algorithms includes

- the formulae to calculate the *Eps* value used in the measure algorithms (Equation 6.1 and Equation 6.2)
- the formulae to calculate the number of features to retain and length of network to retain used in the transformation algorithms (Equation 6.3 and Equation 6.4)
- the procedural knowledge that is the control flow of the algorithms.

The knowledge embedded in the algorithms represents less of a problem because if the characteristics of the algorithms are sufficiently represented in the ontology then a lack of knowledge of their internal behaviour will not affect how they are utilised by software systems and a natural language description will satisfy the curious human.

Of the knowledge embedded in the mapping engine it is unlikely that all control knowledge could be made explicit and formalised in the ontology but it is not clear how much should be. It could be argued that the control knowledge is specific to the implementation and that there is no advantage in sharing it.

In addition to considering the location and representation of knowledge it will be useful to consider what components of the on-demand mapping system should have access to the ontology. For example, rather than allow the measure and transfer algorithms to interact directly with the ontology, it could be stipulated that any communication should always be via the *Mapping Engine* (Figure 6.27). The latter approach ensures that if significant changes are made to the ontology then only the Mapping Engine need be updated. It should be stressed, however, that a change to the ontology does not always require a change to the application that uses it. For example, an early version of the ontology included a generic *Selection* operator. As the prototype was developed it was realised that the *Selection* operator was too general and subclasses of it, such as *Selection by Attribute* and *Pruning* were added. This refinement of the ontology did not require any changes in the mapping engine. This demonstrates the flexibility of the ontological approach; the ontology can be extended by adding new operator and algorithm subclasses and more generally by adding new properties to existing classes without requiring a change in the Mapping Engine. This is a major advantage of formalising the domain knowledge in an external ontology.

7.2.2 Refining the ontology design

The difficulties experienced in designing the ontology were raised in section 5.8. These difficulties can be grouped in the following categories

- should a concept be represented by a class or individual?
- in which class should a property be represented?
- is a particular concept necessary?

The first of these is illustrated by the discussion of whether the *Geometry* concept should be modelled as a class or individual (section 5.4.2). There are a number of concepts where the individuals are not easily comprehended. For example, there is a *symptom* class but it is not obvious what its individuals represent. An individual symptom could represent a particular symptom of a Feature Collection and therefore only exist when the model is executed.

The second category concerns the problems in determining which particular attribute (or item of knowledge) belongs to which concept. For example, there are conflicts between the concepts of an operator and a transformation algorithm (section 5.3.1). It might not be necessary to specify the input geometry of both operators and algorithms. There are similar conflicts between feature *types* and feature *collections* (section 6.2.2). These problems might not be resolved until the ontology is tested with further use cases.

The final category is concerned with whether a particular concept is necessary or not. An ontology class can be defined as a set of individuals sharing the same characteristics (Hart & Dolbear, 2013). There are currently no individuals in any of the *Operator* subclasses (Amalgamation, Aggregation, Collapse etc.) and it might be concluded that the concept of an operator is redundant and the algorithm concept is sufficient to model the process of generalisation.

One solution could be to regard the operator class as similar to an *abstract* class in the Object-Oriented (OO) model, where an abstract class is one that cannot instantiate objects. In the OO model an object is instantiated from a class. However, in OWL an individual can exist separately from the class. Individuals are not instantiated and there is no concept of an abstract class in OWL.

The necessity of encapsulating the concept of the operator was justified earlier (section 5.3.1) as an aid for human comprehension based on the fact that the operator was a key concept in the literature and practice of generalisation. Alternatively, the *Transformation Algorithm* class could be removed and the implementations of the algorithms become the individuals of the *Operator* subclasses.

It could also be argued that the lack of individuals for the Operator subclasses has not stopped the competency questions from being answered satisfactorily. Whatever the solution, these difficulties question the assertion by Davis et al. (1993) that a knowledge representation such as an ontology acts as a surrogate for tangible and intangible objects equally well.

These problems are partly due to the intangible nature of the generalisation domain and partly due to the inexperience of the designer. It would be hard to argue against the assertion of Noy and McGuinness (2001) that “*there is no one correct way to model a domain*”. However, the implementation uncovered one aspect of the ontology design that does require further deliberation and its resolution could help answer some of these apparent problems.

In OWL inference involves determining which class a particular individual belongs to. It can be argued that the interactions with the ontology currently involve too many queries and not enough inference. Some of the Manchester OWL syntax queries seem overly long and complex; for example, the query used to determine a spatial relation between two feature types described in section 6.6.2. The emphasis of the ontology needs to be altered to focus on classifying individuals. For example, rather than query the ontology to determine an appropriate measure algorithm for a particular feature collection (FC) it should be possible to infer the appropriate measure algorithm. This will require a redesign of the ontology. A class of feature collections that require a point density measure algorithm could be defined, for example, and then inference used to determine whether a mapped FC individual is in this class based on the attributes of the FC. If it is then it can be concluded that the FC can be assessed by the particular algorithm.

In summary, to get the full advantage of using the concept of ontologies and inference there needs to be a change of emphasis from *general classes* to *specific individuals*; in particular the focus should be on classifying features collections as requiring a particular measure algorithm, as requiring a particular operator and requiring a particular transformation algorithm. It is relatively easy to understand what the individuals in a FC class represent and it could be argued that if the focus of the ontology was on classifying feature collections then the lack of individuals for classes such as Operator is less important than first thought.

As well as shifting the focus of the ontology there is also a deficiency in content; the ontology is still too focussed on geometry, which can be seen by the OWL queries in Figure 6.30 and Figure 6.31. Although geometry is important it needs to be less prominent in the process of selecting operators. This implies that the ontology requires greater representation of the semantics of generalisation. One aspect of the ontology that requires more consideration is user requirements.

7.2.3 User requirements

The specification of user requirements was identified as necessary for an on-demand mapping system but ruled as out of the scope of this research due to the size of the topic. Despite this, the formalisation of user requirements cannot be entirely ignored in a discussion of on-demand mapping and can, in fact, help with the selection of appropriate operators.

When congestion is identified in the accident features the ontology suggests three operators that would potentially resolve the condition; *selection by attribute*, *amalgamation* and

aggregation (sections 5.7 and 6.8.2). This could mean that the ontology lacks refinement. All three operators are valid in that they remedy the condition (although the effectiveness of the *selection by attribute* operator is dependent on the source data) but there might be a need for other criteria, not currently defined in the ontology, that influence the selection of operators. All three operators vary in their outputs; *selection by attribute* retains the concept of individual accidents, and their distinct details, rather than abstract them into clusters, *amalgamation* retains the geography of the cluster and *aggregation* provides the best indication of the numeric size of each cluster. So there is scope for refining the operator definitions by including these characteristics.

Such a refinement of the operator definitions could be linked to the requirements of specific user *roles*. A number of typical users have already been defined as part of the use case (Table 5.1). For example, the role *road engineer* might require the *preservation of spatial accuracy* and *preservation of feature detail*. In this case an operator, such as *selection by attribute*, that does not change the location of (accident) features and a road network portrayed as area features rather than line features would be preferable. Representing these requirements in the ontology would help guide the choice of operator.

Currently, when more than one operator is suggested by the ontology the prototype prompts the user to select one from a drop-down box (Figure 6.32). The prototype could be adjusted such that the preferred operator, its identification now aided by the user requirements, could be highlighted but the user given the final choice. However, the non-expert user cannot be expected to make an informed decision based solely on the names of operators. One solution would be to provide visual clues to the effects of each operator (Figure 7.1).

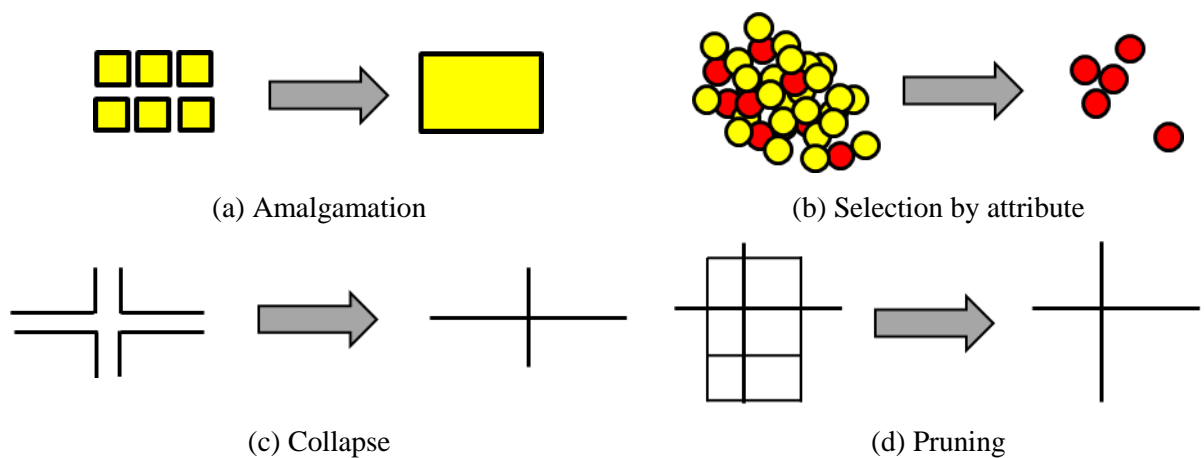


Figure 7.1 Example icons for generalisation operators

In addition to determining appropriate generalisation operators the concept of the *role* could be used to guide the selection of feature types to map. For example, the addition of a school would improve the map generated for the *parent* role (Figure 6.35). This would involve defining the role in the ontology and then defining the feature types relevant to that role (Figure 7.2).

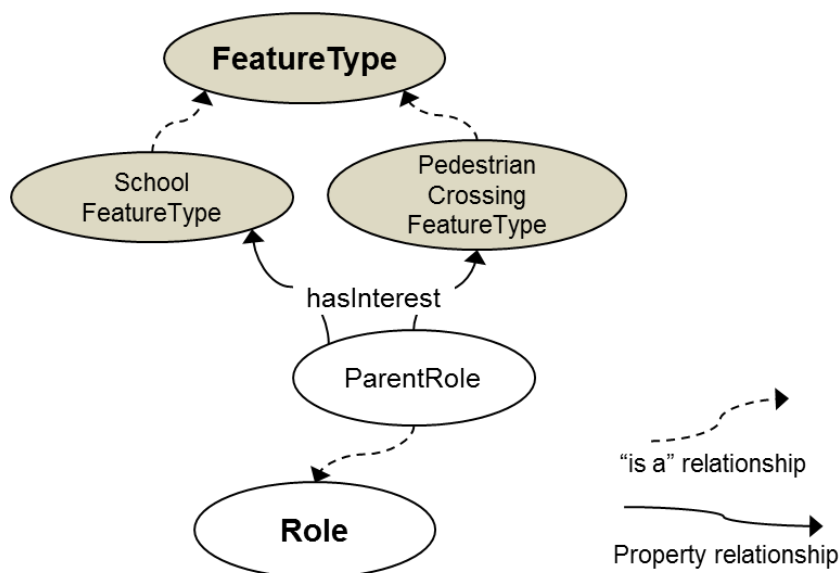


Figure 7.2 The relation between feature type and role

One of the main requirements of an on-demand mapping system is an ability to map features at multiple scales and it is necessary to reflect on the concept of scale, and how it is represented in the ontology, in more detail.

7.2.4 Scale

The concept of scale is not *explicitly* defined in the ontology. Given that many existing algorithms are designed to generalise features over a specific scale range, for example 1:10K to 1:50K, this may be a significant omission. The process of assessing the geometric conditions of the mapped features and then identifying the solutions is a *bottom-up* approach (as discussed in section 4.4.3) and is characteristic of the McMaster and Shea model. A consequence of this is that, in the current implementation, the same operators are suggested for the same conditions whatever the scale and whatever DoG is calculated. Is it possible, or desirable, to incorporate the concept of scale explicitly and allow a *top-down* approach, where the features and their representation are selected based on the current scale?

The attraction of the top-down approach can be seen if we consider the accidents in the Greater Manchester area over the same time period as the use case but covering an area of

approximately 1200 km² compared to the 18km² of the use case data (Figure 7.3). The calculation of a measure of feature density for this number of accidents (155,919) was beyond the simple measure algorithm developed because of an upper limit on the size of the matrices used by the algorithm (section 6.3.1) but even with a more sophisticated algorithm the measurement would still be time-consuming. The application of a top-down approach in this example is therefore attractive.

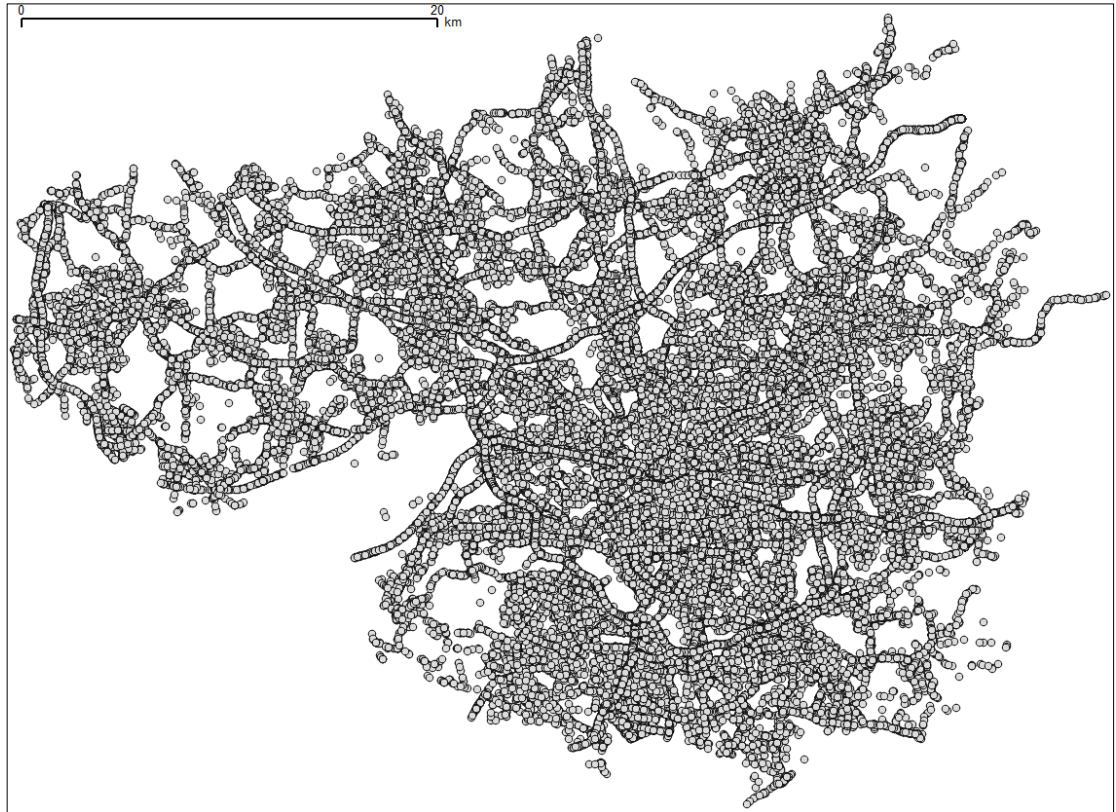


Figure 7.3 Road accidents in Greater Manchester 1994 to 2008

Such an approach could be implemented in a multi-scale on-demand mapping system but it would require the definition of a set of scale ranges (e.g. 10K to 20K, 20K to 30K) and the enumeration of the features that were relevant in each range and how they were best represented. However, such an approach is akin to defining procedural knowledge and comparable to the rule-based systems that were dismissed earlier (section 2.4). It is necessary to encapsulate the characteristics of a particular scale change and then *infer* the appropriate action as has been done with the use case.

The issue of whether an operator was applicable at a particular scale was raised when the application of the displacement operator to accident features at small scales was rejected but

the reasons *why* were not encapsulated in the ontology (section 5.2.1). The remainder of this section will consider ways of doing this.

Displacement is one of the few operators where there is a consensus on its meaning (Figure 2.6) although it is hard to implement (Regnauld & McMaster, 2007). As discussed earlier (section 5.2.1), the use of displacement is not applicable to this use case as it is difficult to apply in areas of high feature density (Regnauld & McMaster, 2007) and it is more relevant when a small number of topographic features are involved. For example, a road and a railway line, and the coastline they follow, will tend to coalesce when portrayed at a small scale (Regnauld & McMaster, 2007) (Figure 7.4). Topology should be respected during displacement so the railway line stays between the road and the coastline.

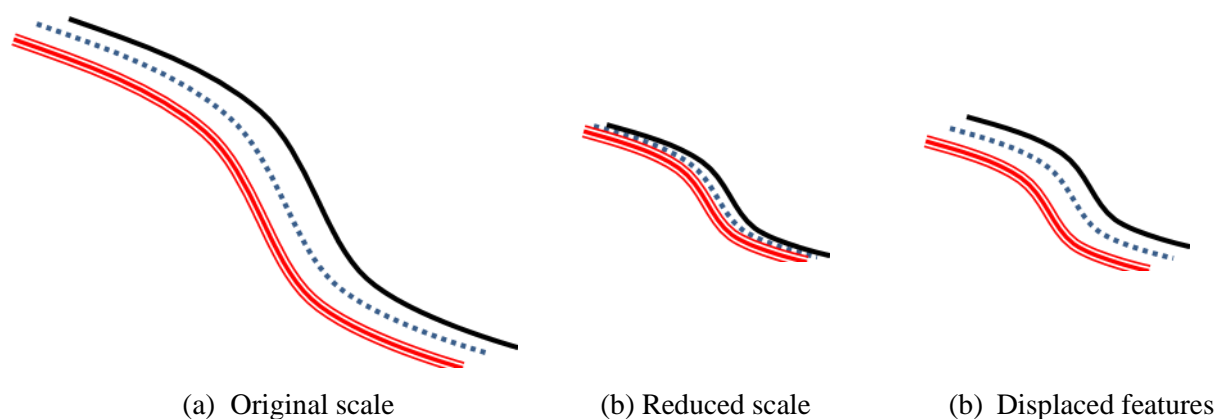


Figure 7.4 Applying displacement

Displacement is inappropriate at a small scale where many features are mapped and hundreds, potentially thousands of features would have to be displaced to reduce congestion (Figure 7.3). The ontology must represent the fact that this operator is not applicable to the use case. One option would be to define a feature density threshold, above which the application of displacement would be forbidden but this approach seems too similar to the application of a rule. There is however, an alternative, semantic, approach.

McMaster and Shea (1992) make a distinction between the geometric conditions *congestion* and *coalescence*. Whereas the former is the result of mapping too many features than is appropriate for the particular scale, the latter is defined as “... *a condition where features are closely, partially, or completely in juxtaposition in their map or geographic coordinate locations*” (p44) and is regarded as a result of the representation of the features rather than their quantity. It could be argued that the features depicted in Figure 7.4b are suffering from coalescence and not congestion. If the ontology defines displacement as a remedy for coalescence but not congestion then it will not be suggested as a solution to accident

congestion. This approach will prevent operators such as amalgamation that resolve congestion, from being applied to situations such as that in Figure 7.4b.

It is, therefore, not then the scale itself but the number of features that are mapped (which is a *consequence* of the scale) that prohibits the use of the displacement operator (Figure 7.3). For example, the London Underground map originally designed by Harry Beck (Figure 7.5) contains a great amount of displacement and is at a similar scale to the Greater Manchester accident map. This is why the ontology can manage without an explicit definition of scale. Rather than capture in the ontology the knowledge that a particular algorithm is suitable for a particular target scale it is necessary to capture the *reasons* for its suitability thus allowing that knowledge to be repurposed for other situations. This is the “new expressions from old” (section 3.5).



Figure 7.5 London Underground map (fragment) © Transport for London

If this solution is adopted it will still be necessary to develop one or more measures for *coalescence* that are distinctive from the measures of high feature density that identify congestion. One possibility would be to assert that coalescence is only applicable to features of different types such as those in Figure 7.4. However, the routes mapped in Figure 7.5 are of the same feature type and would have suffered from coalescence so perhaps it is necessary to additionally consider the *number* of features. In the implementation of the Mapping Engine, *four* features was set as the lower limit for the size of any cluster that suffered congestion (section 6.3.1), so coalescence could be defined as involving *three* or fewer features. However, this seems arbitrary, and again reminiscent of a rule, and the distinction between congestion and coalescence requires further consideration.

To what extent is the McMaster and Shea model suitable for on-demand mapping? For mapping familiar feature types (building, roads and rivers, for example) at a limited range of scales then the constraint-based model is the dominant model (section 2.5). However, when mapping unfamiliar feature types at multiple scales then the concept of measuring the condition of the features before deciding when to generalise has worked with the, admittedly limited, use case.

However, the model's focus is on the *cartographic* rather than the *geographic*; it focusses on resolving *geometric* conditions such as congestion and imperceptibility. It will be necessary to update it to focus on resolving *semantic* conditions such as the relationships between feature types. For example, not only “we can not see the accidents legibly at this scale” but also “the relationship between roads and accidents is not maintained at this scale”.

As well as exploring the appropriate scales for different operators it is also necessary to consider which feature types should be mapped at particular scales. This can be considered more effectively if we look beyond the current use case.

7.3 Beyond the use case

The previous section discussed *scale* in the cartographic sense; here it is discussed in the context of the *scalability* of the software solution. The system worked for the road accident use case (with the qualifications discussed in chapter 6) but it is important to consider how easy it is to apply the solution to other use cases. This can be measured in terms of the extra work required; the extent to which the ontology and the on-demand mapping system software have to be modified to accommodate other use cases. If two new feature types can be mapped by simply adding two new classes to the ontology and describing their relationship then the solution could be said to be scalable. However, if “significant” changes to the ontology or mapping system have to be made then the solution cannot be acknowledged as such. To assess scalability three scenarios, with increasing variance from the original use case, are discussed

1. extending the current use case to include new feature types
2. examining another use case but in the same transport theme
3. examining another use case in a different theme.

7.3.1 Extending the current use case

The use case currently maps two feature types; accidents and roads. At a large scale the addition of bus stops and traffic lights would provide useful context (Figure 7.6). For

example, if it can be seen in Figure 6.33 that the central T-junction has a relatively high number of accidents but at the other two T-junctions there are few or no accidents. The mapping of additional features might help to explain why.

It was suggested that the decision to map schools could be influenced by the *role* concept (section 7.2.3). Here, the traffic lights and bus stops feature types, can be defined in the ontology as subclasses of a road transport feature type class (Figure 5.11) and since they share a superclass with road accidents the mapping engine could suggest them to the user as feature types that provide context.

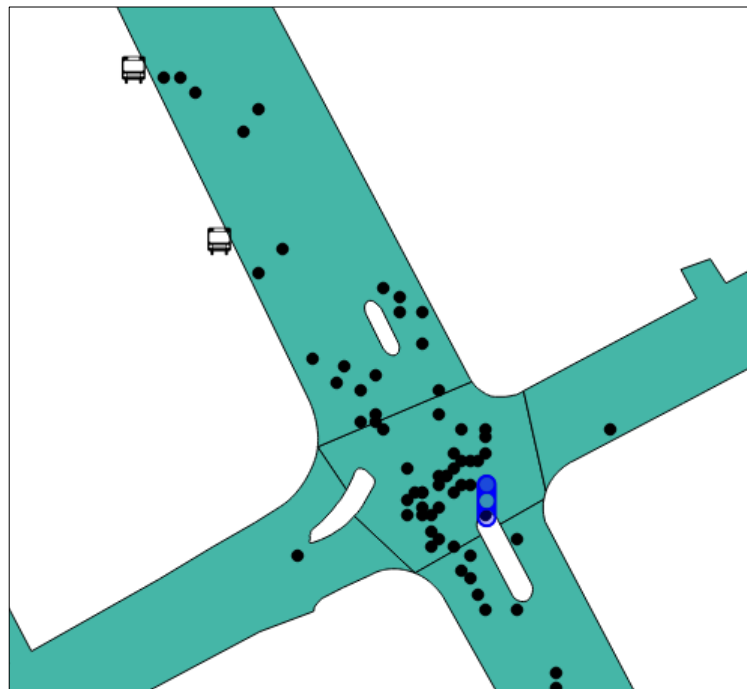


Figure 7.6 Accidents at a junction with traffic lights¹⁵ and bus stops

At a smaller scale both of these datasets will suffer from congestion as the accident dataset did and, since they are both point features, the ontology would suggest *selection by attribute*, *amalgamation* and *aggregation* as remedies. However, selection by attribute is not likely to be an option since neither dataset is likely to have an importance attribute; one bus stop is unlikely be more important than another. *Amalgamation* and *aggregation* are possible but not appropriate, since there is no concept of a “hot-spot” of traffic-lights or bus stops as there is with accidents or crimes. *Amalgamation* and *aggregation* are appropriate for other physical features such as buildings or trees since there are higher level collective concepts that represent these features; such as a built-up area, village and town for buildings and a forest for

¹⁵ The single traffic light symbol represents a *set* of traffic lights at a junction.

trees (Figure 7.7). Why are there no equivalent higher level concepts for bus stops and traffic lights and how can the ontology be designed so it can be inferred that these operators are not suitable for these feature types at small scales?

A traffic light may be described as a property of a node (road junction) on a network; without the node and the network there is no traffic light. Similarly, a bus stop can be regarded as an access point to a bus route network. Although both feature types are material objects, neither has an independent existence and therefore no higher level abstraction. This is in contrast to a *hill*, for example, which has multiple functions - a view point, part of walking route, a flood-free location - and can exist independently from other concepts.

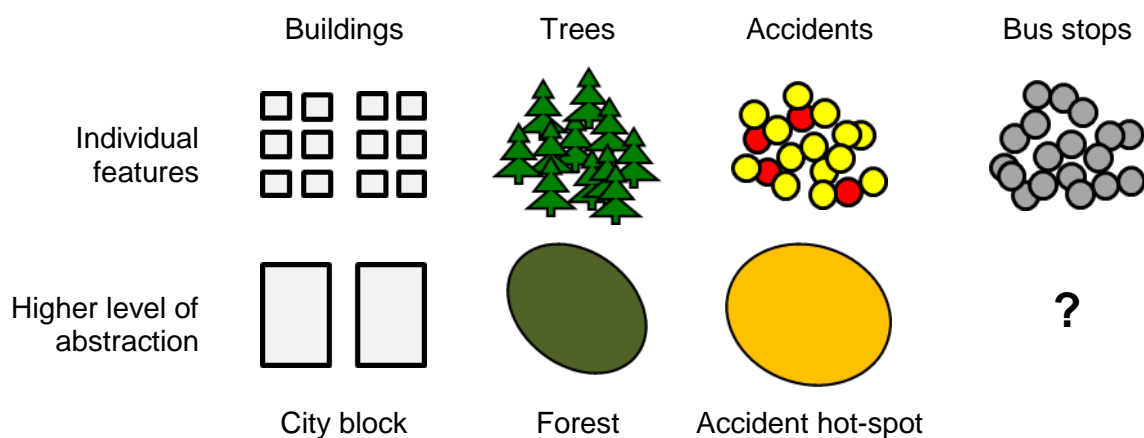


Figure 7.7 Levels of abstraction for different feature types

Table 7.1 lists a number of feature types and determines whether they have a higher level collective concept or not. It can be concluded that those that do not should not be amalgamated or aggregated, but this needs to be modelled in the ontology.

Feature type	Function	Higher level collective concept
Manhole cover	Access point to a network	None
Bus stop	Access point to a network	None
Traffic light	Control flow at a network node	None
Tree	Multiple	Copse, wood, forest
Building	Multiple	Block, village, town
Hill	Multiple	Range

Table 7.1 Higher level collective concepts for different feature types

Firstly, Bertin's concepts of *structural* and *conceptual* generalisation (discussed in 5.2.1) can be used to classify the different operators. Operators such as *amalgamation* and *aggregation*, that change the level of abstraction of the features, can be classed as conceptual operators and those that do not, such as *selection by attribute* and *pruning*, are classed as structural operators (Figure 7.8). The *NetworkObjectFeatureType* class is asserted to forbid conceptual generalisation and its subclasses will inherit this property. Another possible subclass of *NetworkObjectFeatureType* might be *man-hole covers* (used to access a sewerage network). The *forbids* concept was introduced in section 5.5.2 and here it is used to prevent any *NetworkObjectFeatureType* from being amalgamated or aggregated. Therefore, as scale reduces, such features will be mapped until the scale is small enough for them to exhibit congestion. At this point, unless *selection by attribute* is possible, the features will no longer be mapped.

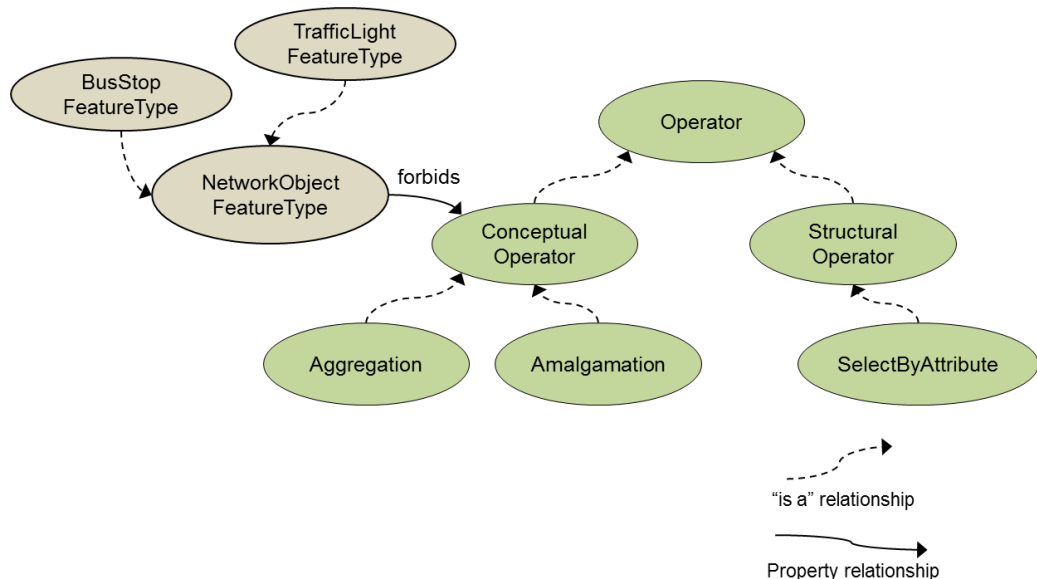
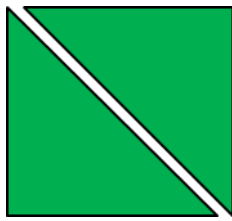


Figure 7.8 Conceptual and structural operators

There is however, at least, one problem with this solution in that amalgamation does not always cause a change in the level of abstraction. Consider the two features in Figure 7.9a. If they represent two buildings separated by a road then when they are amalgamated they will represent a new concept, a building block or built-up area. However, if they represent two areas of woodland separated by a track then following amalgamation there is merely a larger area of woodland.



(a) original features



(b) features at reduced scale



(c) amalgamated features

Figure 7.9 Amalgamating two features

This problem could be possibly solved by defining the concepts of *countable features* (buildings) and *non-countable features* (woodland) in the ontology. This example highlights the importance of semantics to generalisation and the benefit of capturing those semantics in an ontology.

To summarise, the extension of the use case to include the mapping of bus stops and traffic lights requires the following changes to the ontology

- the addition of the *bus stop* and *traffic light* feature types (Figure 7.8)
- the definition of a number of *roles* and their properties such as their relation to different feature types (Figure 7.2)
- the definition of the relationships between the new feature types and the *road* network feature type (similar to section 5.5.6). This is perhaps less important than the relationship between the road and accident feature types since, at the scale where the roads require pruning, the new feature types are not likely to be mapped.

7.3.2 Mapping bus routes

The second proposed new use case involves the same theme, road transport, but considers the mapping of bus routes at different scales. As with the original use case, three different scales are considered.

At a relatively small scale (approximately 1:300K) the extent of the bus route network in Greater Manchester is depicted but the map is not particularly useful (Figure 7.10). What would be useful at this scale would be to limit the map to the major cross-region routes. This could be achieved relatively easily by pruning the network and retaining the longest strokes. However the strokes need to be determined by the route (service) number rather than the angle between road segments. Since the routes are not to be pruned with respect to another feature, an algorithm based on the first version of the road pruning algorithm developed in the implementation (section 6.6.2) is required. This revised algorithm would take as input an

attribute similar to the importance attribute required by the *selection by attribute* algorithm (section 6.5.1), which would define the main factor in determining whether two segments were part of the same stroke. In this example, the “*pruning attribute*” would be the route number. The use of the algorithm would be triggered by the detection of congestion in the dataset (Figure 7.10).

A modification to the ontology would be required to ensure the selection of the new algorithm for those network feature collections with an importance attribute specified. The ontology would also have to be extended to differentiate between *simple* pruning and pruning a network with respect to another feature collection. The latter would require the existence of a relationship between the feature collection to be pruned and another mapped feature collection such as exists between roads and accidents. At this scale a schematic map (similar to Figure 7.5, for example) might be more appropriate (Mackaness & Reimer, 2014).

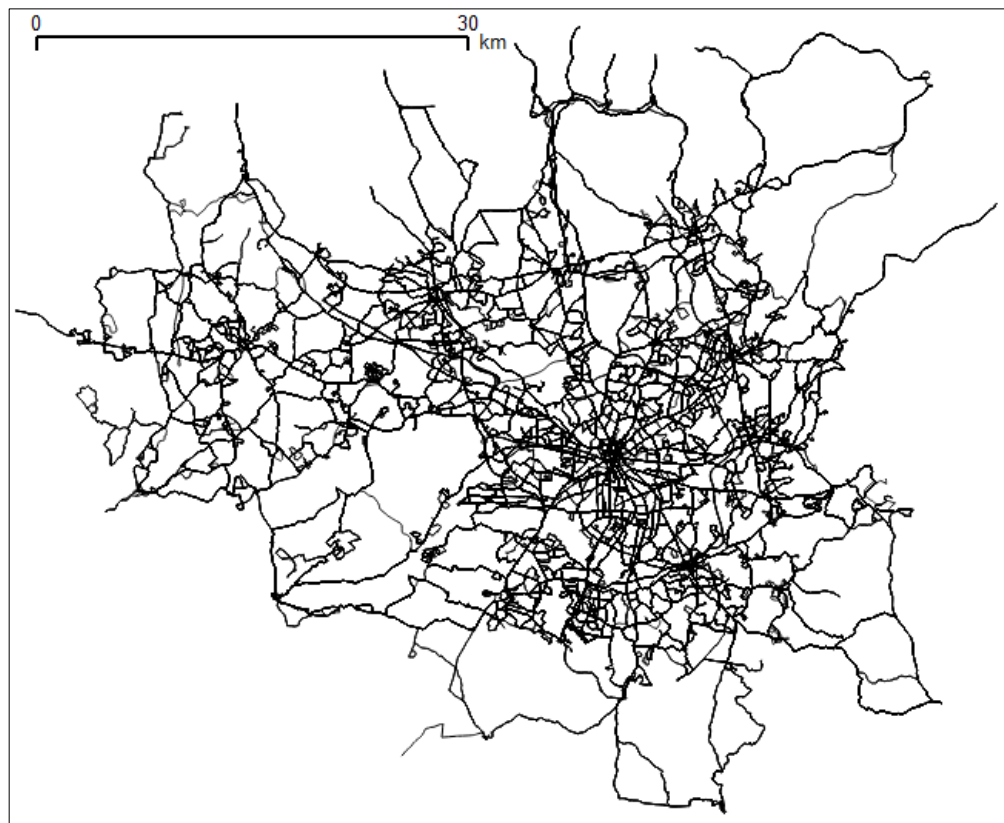


Figure 7.10 Bus routes in Greater Manchester

At a medium scale the requirement might be to show a particular route at a neighbourhood level (Figure 7.11). At this scale (approximately 1:30K) the local road network would provide some context. As can be seen the road network is not congested and the algorithm that measures the density of road line features (section 6.3.2) is unlikely to identify any

problem features thus it could be concluded that no generalisation is required. However, there is a lot of unnecessary detail and perhaps only those strokes that intersect the network should be mapped. This is a situation where the focus of the McMaster and Shea model on the existence of geometric conditions to determine when to generalise has its disadvantages. In this case there is a semantic rather than a geometric reason to generalise and remove those road segments that do not provide context for the bus route.

The semantic relationship between accidents and roads was ultimately expressed as a number of spatial relationships (section 5.5.6) and this will be the case with the relationship between the bus routes and the road network: the road segments mapped will be those that intersect a buffer around the bus routes. This questions the need for defining the semantic relationship between features if they are also going to be expressed as spatial relationships.

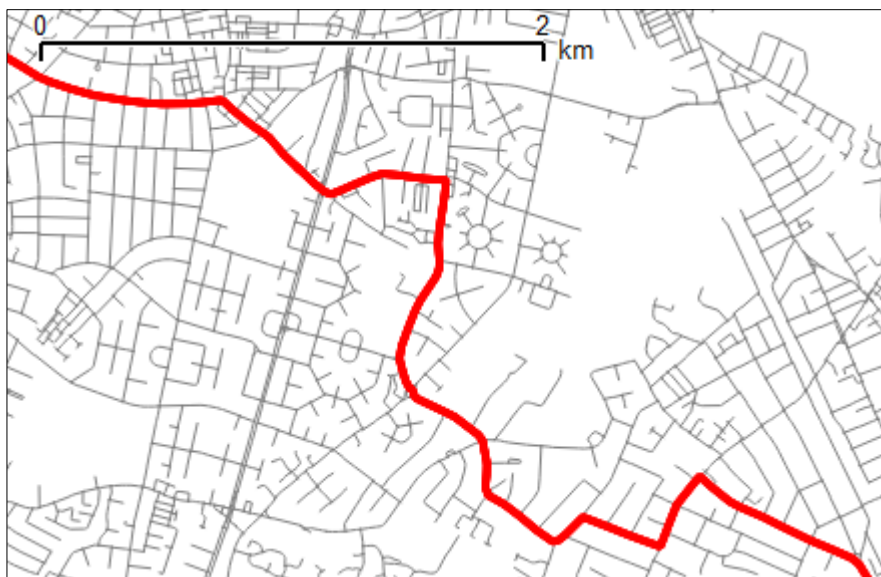


Figure 7.11 A bus route (in red) at the neighbourhood scale mapped with the road network

Firstly, the semantic relationships in the two use cases are different. The mapping of the road network provides a context for the bus routes and aids understanding of the bus routes and will, for example, aid the bus driver in learning a new route. The mapping of the road network with the road accidents does more than provide context; without the road network the map is meaningless as the road features form part of the definition of a road accident.

Secondly, if we define the relationships at the higher, semantic, level they can be repurposed. If we assert that the mapping of the major roads and those minor roads that intersect a buffer around a bus route “provide context” to that bus route we can reuse that definition of “providing context” in other situations with other feature types.

Finally, if we consider geometry alone, then there is no reason to generalise the road network in Figure 7.11, despite the fact that if it is generalised then redundant data will be removed from the map.

At this scale *landmarks* might provide some context. Landmarks are a complex concept as they do not form a single feature type and can be natural (a hill, the bend of a river) or man-made features (a building, a highway) (Richter & Winter, 2014). As with the road network only those landmarks that provide context to the bus route are relevant and it is important to determine which landmarks should be mapped at which scale. A small shop would not be considered a landmark at a small scale (Figure 7.10) but at a larger scale (Figure 7.11) it might provide a useful addition to the map. Depending on the scale almost any tangible feature can act as a landmark and it is necessary to define a hierarchy of landmarks (Grabler et al., 2008).

If the discussion is limited to buildings as landmarks then a hierarchy would need to be established to ensure that only the most important buildings are mapped at a small scale. The relationship between the bus routes and the landmark buildings is weaker than that between accidents and roads. The relationship might be of the form “landmark buildings *provide context to* bus routes” or “landmark buildings *aid navigation of* bus routes”. Another approach is to create a new role, *traveller*, say, and define that role as requiring landmarks. Either way, the semantic relationship can be expressed as one or more spatial relationships such as “landmark buildings *are near* bus routes”. This will allow for the selection of only those landmarks that are relevant to the particular route.

The ranking of the buildings as landmarks is a complex decision, involving a number of factors such as building function, height and age. However, if building *area* alone is considered then each building can be classified in, say, 10 categories, where buildings in category 1 have the largest footprint and the highest importance. This attribute would be then flagged as the importance attribute for the feature collection in the ontology and the collection would then be inferred to be a *RankedFeatureCollection* (section 5.5.1). This would mean that the feature collection would be eligible for *selection by attribute*, and only the most importance landmarks mapped (section 6.5.1).

Assuming the generalisation of the building features is triggered by congestion then *amalgamation* operator (section 5.5.2) will also be suggested by the ontology as a remedy. However, this would effectively destroy the landmarks; therefore a mechanism is required to

prevent this operator from being suggested by the ontology. In this case the features mapped are landmarks rather than buildings and have to be defined as such in the ontology using a *LandmarkFeatureCollection* class, for example. The *forbids* relationship, which was used to prevent the amalgamation of features in a network (Figure 5.22) and the amalgamation of features such as bus stops (Figure 7.8) could be used to prevent the amalgamation of landmarks.

The *forbids* property has now been used in a number of situations and it might be argued that its use is similar to procedural knowledge; for a particular feature type *forbid* a particular operator. This could be seen as being contrary to the aim of asserting as little as possible and inferring as much as possible. However, the assertions involving the *forbids* property have been indirect, passing through intermediate concepts such as a *network* (Figure 5.19 and Figure 5.22) or a *conceptual operator* (Figure 7.8). So we can *infer* the road features should not be amalgamated because we have *asserted* that the road features are part of a network and we have asserted that a network cannot be amalgamated. It may be possible to detail in the ontology *why* a network should not be amalgamated but ultimately all inferences are based on assertions. The aim should be to make the assertions about higher level concepts such as networks.

Bus routes are networks. They do not have formal nodes but the point where they intersect is a virtual node and allows the traveller to switch routes (this might not be in the exact location of the intersect but it will be nearby). Individual bus stops can be mapped along with the route network at a relatively large scale (approximately 1:5K) (Figure 7.12). The mapping of the bus stops was discussed in the previous use case.

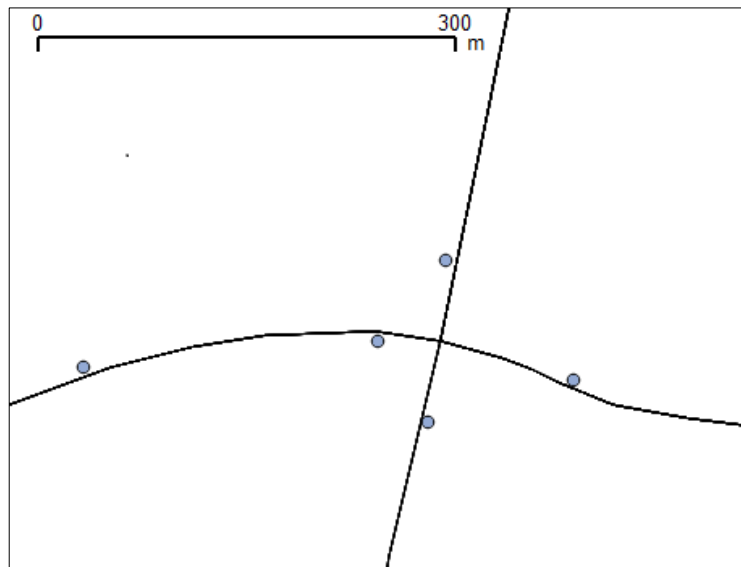


Figure 7.12 Two intersecting bus routes with bus stops

To summarise, the following developments to support the multi-scale mapping of bus routes have been identified

- the development of the basic pruning algorithm to develop strokes based on a specified “pruning attribute” (e.g. route number)
- the modification of the ontology to ensure the selection of the new algorithm for those network feature collections with a pruning attribute specified
- the recording in the ontology of the relationship between the road network and the bus routes
- the addition of a new, *Landmark* feature collection class and the definition of its relation to the bus routes.

Although coalescence was discussed briefly, so far only a single geometric condition, *congestion* has been considered in detail. The following section will discuss the consequences of modelling another condition.

7.3.3 Assessing a condition other than congestion

The original use case concentrated on the single geometric condition of *congestion* in two feature types: roads and accidents. The proposed new use cases considered the mapping of different feature types but still focussed on the congestion of features. To test the model further, this next section will consider another geometric condition defined in the McMaster and Shea model (Figure 3.1), *imperceptibility*; which, here, is defined as the condition that occurs when a feature is portrayed with too much detail for the current scale. In addition to being a different condition from congestion it also operates on individual features as opposed

to the multiple features that congestion applies to. It is therefore a useful test of the ontological approach.

If we consider the rather complex building outline in Figure 7.13a (the supermarket portrayed in Figure 1.2a), we can see that at a reduced scale there is too much detail and the building outline is hard to determine (Figure 7.13b). The feature would benefit from simplification (Figure 7.13c) but first, “too much detail” needs to be quantified. When considering *individual* features, Stigmar and Harrie (2011) state that legibility is influenced by *object complexity*, which is determined by the shape and size of the features, and list several relevant measures; including object size, line segment size, angularity, line connectivity, and polygon shape.

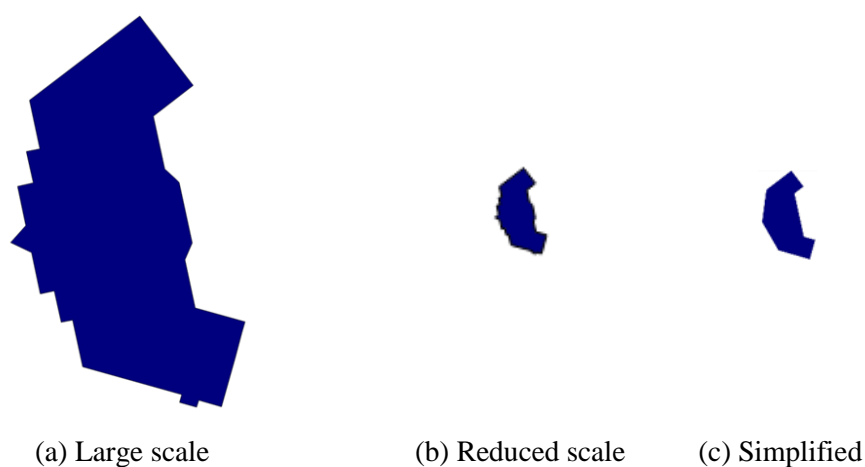


Figure 7.13 A building represented at two different scales

Harrie and Stigmar (2010) define a measure for the number of object points that simply sums the number of points used to define each mapped feature. This can be adapted to individual features by defining a *vertex density* as the number of points used to define the feature divided by its area¹⁶. However, the threshold value that determines whether the feature is imperceptible must be scale dependent, in the same way that the *Eps* value was (section 6.3.1), since the same feature at a larger scale will not suffer from imperceptibility. However, this measure is not without disadvantages. Consider the two buildings portrayed in Figure 7.14a. They both have the same area and are defined by the same number of points and therefore have the same vertex density but at a reduced scale their perceptibility differs (Figure 7.14b); it is not clear whether *Building 2* is, in fact, two separate buildings.

¹⁶ For linear features, the number of points divided by the line length would provide an equivalent measure. For point features, of course, the condition is not relevant.

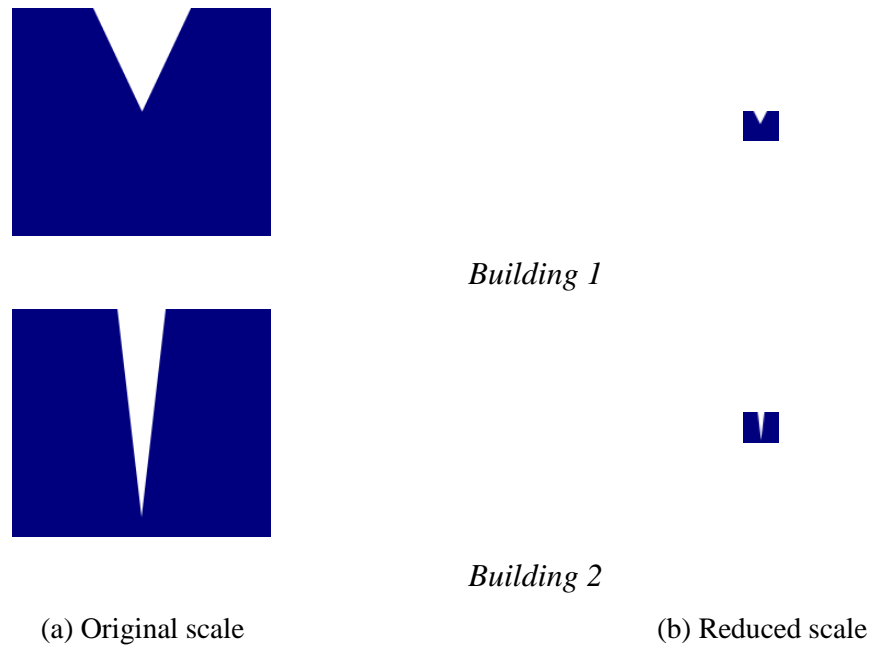


Figure 7.14 The effect of building shape on perceptibility

The *polygon shape* measure calculates the ratio between the area of a feature and the area of its convex hull and can be used as an indicator of the complexity of a polygon feature (Stigmar & Harrie, 2011). However, in this example (Figure 7.14) its application would again result in the same value for both buildings. So a further measure might need to be applied, such as *angularity*, which considers the changes in the direction of a line that defines a feature (Stigmar & Harrie, 2011). In this case Building 2 would be identified as having a high angularity.

Currently it is possible to define a number of measures in the ontology that will identify symptoms and thus identify a particular condition but there is no mechanism to *combine* measures to identify a condition. A process that is similar to Multi-Criteria Decision Making (MCDM) (Longley et al., 2005) is required, where the existence of a condition is defined by the sum of its symptoms. This can be modelled in the ontology. For example, if the class *ImperceptibleFeature* is defined as

```
(hasSymptom some HighAngularity) and (hasSymptom some HighShapeComplexity)
and (hasSymptom some HighVertexDensity)
```

then any feature, and only those features, that possessed all three symptoms would be classed as an *ImperceptibleFeature*. Alternatively, the class could be defined as

```
(hasSymptom some HighAngularity) or (hasSymptom some HighShapeComplexity)
and (hasSymptom some HighVertexDensity)
```

then only those features that had the *HighVertexDensity* symptom and either or both of the other symptoms would be classed as an *ImperceptibleFeature*. So it is possible to model a combination of symptoms that defines a condition.

If the condition of imperceptibility is identified in the building feature (Figure 7.13b) then it needs to be remedied. All three of the symptoms described above can be remedied by reducing the number of vertices in the features. This can be implemented by the *Simplification* operator, which in turn can be implemented by any number of algorithms. Consider the most well-known of these, the line simplification algorithm of Douglas and Peucker (1973). The DP algorithm is not sufficiently sophisticated for building simplification (Guercke & Sester, 2011) but has been selected here for its simplicity. It has a single parameter, *tolerance* and a relationship between this parameter and the DoG needs to be determined. Previously the DoG values have been derived from the ratio of *problem features* (as identified by the measures) to *mapped features* (Equation 4.1). Here, however, we are examining how to generalise a single feature. So it will be necessary to combine the results of the applied measures to determine a value for the extent of the imperceptibility of each feature. With the current system design the knowledge (specifically the equation) to do this will be embedded in the Mapping Engine. A further equation that converts the imperceptibility value into a DoG is required. Finally the line simplification algorithm has to convert the DoG into a tolerance value. The higher the DoG is then the higher the tolerance value and the greater the simplification (Figure 7.15). There is not necessarily a linear relationship between the tolerance and the number of vertices removed. For example, tolerance values from 4 to 9 all removed the same number of vertices (Figure 7.15e). This is similar to the non-linear behaviour of the selection by attribute operator (section 6.5.1) and demonstrates that the DoG concept requires further refinement.

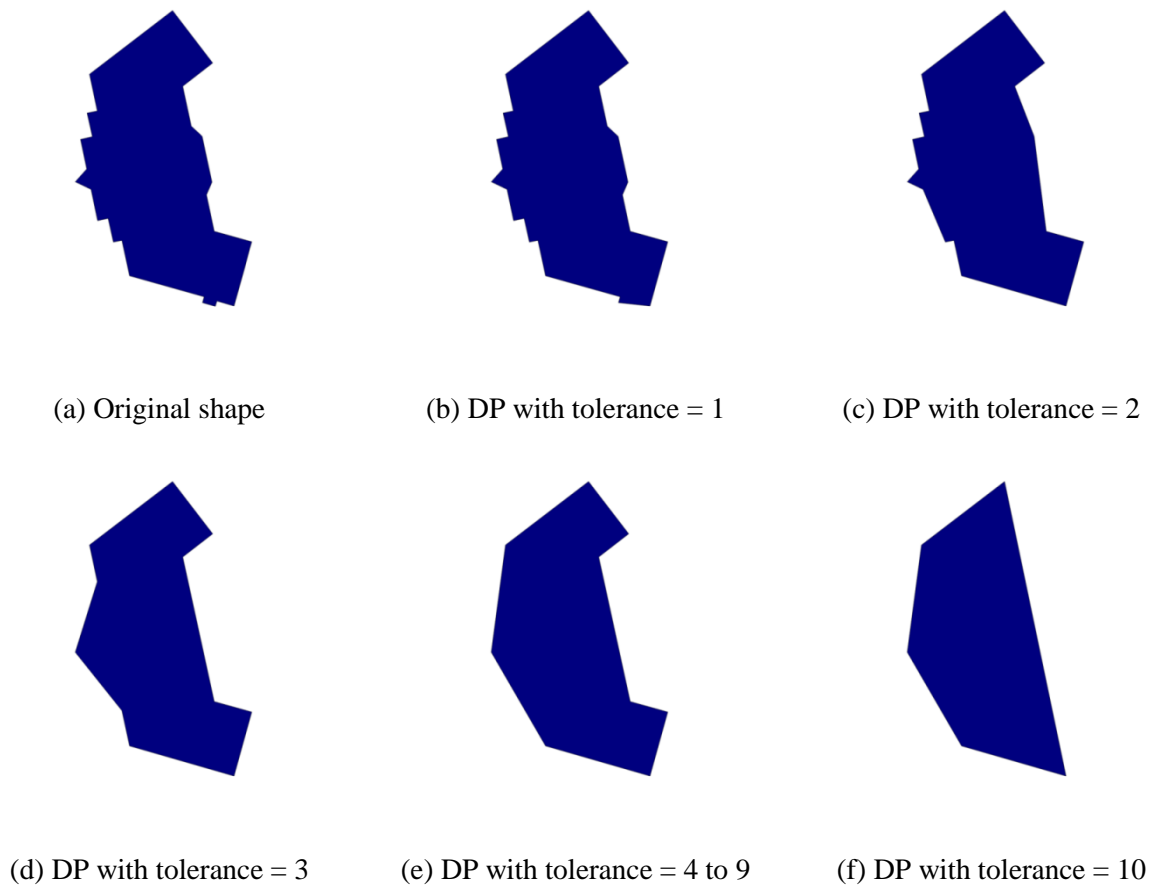


Figure 7.15 The application of the Douglas-Peucker (DP) algorithm with different tolerances

To summarise, the implementation of this use case will require greater changes to the system than those discussed in sections 7.3.1 and 7.3.2. Changes are required not only to the ontology but also to the mapping engine as a result of introducing a new geometric condition to the system. The changes include

- the definition of the *imperceptibility* condition and its symptoms in the ontology
- the development of a number of measure algorithms to assess the new symptoms
- the definition of the new measure algorithms in the ontology
- the inclusion in the Mapping Engine of the equations to
 - calculate an imperceptibility value for a feature
 - calculate a value of DoG based on imperceptibility
- the development of one or more feature simplification algorithms that will utilise the DoG value

A means of representing equations in the ontology would ensure that no changes to the Mapping Engine would be required; only changes to the ontology and the development of new algorithms.

7.4 Summary

This chapter has looked beyond the original use case and considered the changes to the ontology and the Mapping Engine that would be required to apply the on-demand mapping system to other use cases. Without implementing the new use cases it is not possible to be certain, but the only major changes are when the system is required to take account of a new geometric condition; in particular changes are required to the Mapping Engine. However, these changes are finite as the number of conditions are limited; McMaster and Shea list only six in their model (Figure 3.1).

The adaption of the system to new use cases will require more measure and generalisation algorithms. In the original use case a number of algorithms were developed from new rather than using existing algorithms. This was because of the need to incorporate the DoG concept. It is neither practical nor desirable to develop a complete set of new generalisation algorithms for the on-demand mapping system and some method of integrating the DoG concept with existing algorithms needs to be found. One possible option is an adaptation of the *translator function* concept (Touya et al., 2010) to allow the DoG to be translated into the parameter values of the generalisation algorithm. This will not be a simple task since the variety of generalisation algorithm parameters, even among algorithms implementing the same operator, is wide. The work of classifying algorithm parameters, started in section 4.4.1, will need to be extended to support this task.

In contrast, the changes to the ontology required by the new use cases were relatively straightforward. There was no need to alter the essential structure of the ontology to apply the model to different use cases. The changes required were additions to the sub-concepts in the ontology, such as new feature types. The only new top level concepts required were a user *Role* and a *Landmark*.

The ontology can represent either *procedural* or *declarative* knowledge but the intention was to encapsulate declarative knowledge in the ontology (section 3.3). Cartographic *constraints* can be classed as declarative knowledge and it might be asked why constraints were not captured in the ontology, since constraints-based generalisation is a well-used approach. Constraints were ruled out because of the difficulty of defining constraints for all mapped feature types at many different scales and also because constraints do not capture the semantics of generalisation (section 2.5). It is necessary, therefore, to consider whether the alternative knowledge representation, modelling the semantics of generalisation in the ontology in the form of declarative knowledge, was successful.

Most of the assertions about feature types are done at a relatively abstract, level such as a *network* feature collection rather than at the lowest level, such as *road* feature collection (Figure 5.22). So it would only be necessary, for example, to define a *river* feature type and assert that it is a network for it to be generalised in a similar manner to a road network. Or a crime feature type could be created and classified as a point event and mapped in a similar manner to the accidents, but perhaps with different features providing context. So the aim of deriving new expressions from old was met but the implementation of further use cases will be necessary before it can be concluded that the semantic, declarative approach can support on-demand mapping. The ontological approach will still require the representation of a lot of knowledge but if the ontology was built collaboratively then much work could be saved.

8 Conclusions and further work

Following a summary of the work completed (section 8.1), this chapter considers how the ontological approach to on-demand mapping could be extended beyond the McMaster and Shea model (section 8.2). The thesis concludes with a summary of the major achievements (section 8.3).

8.1 Thesis summary

On-demand mapping requires automatic generalisation and an assessment of the difficulties of automatic generalisation was made. A survey was done on how generalisation knowledge had been represented in the past; in software systems and in generalisation operator taxonomies. Previous approaches to on-demand mapping were also investigated (section 2.6).

A case was made for formalising and sharing cartographic generalisation knowledge by using an ontology and then reasoning with that ontology to automatically select generalisation algorithms. The ontology had to be based on a model of generalisation and the McMaster and Shea (1992) model was selected as the most appropriate for on-demand mapping. Because of the lack of an existing methodology specifically to design an *application* (or *task*) ontology, a hybrid methodology, based on a number of existing methodologies was developed (Chapter 3).

To evaluate and make use of the ontology an on-demand mapping system was required and a methodology, based on CommonKADS, was devised for integrating *domain* knowledge (from the ontology) with *control* knowledge. A method was also developed to automatically derive generalisation algorithm parameter values based on the concept of a *Degree of Generalisation* (*DoG*), a measure of the “amount” of generalisation required to resolve a condition. The value for the DoG is a function of the number of *problem features* identified by the geometric condition *measure* algorithms (Chapter 4).

An ontology was built using the methodology and evaluated using the competency questions devised in Chapter 3. A number of use case specific challenges were identified and addressed (section 5.5).

To properly evaluate the ontology a prototype was developed. The prototype was tested at three different scales, each scale representing the requirements of a different type of potential user and satisfactory results were obtained (Chapter 6). A set of generalisation algorithms that employed the DoG as an input parameter was developed but a technique for integrating the DoG with existing algorithms is still required.

Consideration was given to how the ontology could be extended to other use cases (Chapter 7) and it was concluded that the ontological approach was scalable; although those extra use cases, and others, will have to be implemented before this can be stated with authority.

It may be argued, with some justification, that the scope of the thesis is too wide. The scope could have been limited to the design and development of the ontology and its evaluation limited to testing the competency questions (section 3.7.3). This would then have allowed time for the development of a more sophisticated ontology. However, since the aim of the research was to develop an *application* ontology to support on-demand mapping as opposed to an all-purpose *domain* ontology, the ontology could only be properly evaluated by attempting an implementation of an on-demand mapping system. This necessitated the development of a method for automatically generating generalisation parameter values which led to the development of the DoG concept. This in turn led to the need to develop a set of generalisation algorithms that would work with the DoG concept (Most existing road pruning algorithms, for example, have been developed for a particular scale). In addition, on-demand mapping is an immature concept and has not been researched extensively. All of these considerations meant that, by necessity, the scope of the thesis is wide.

The aim and objectives were stated in section 1.5. The main aim was to “*develop an on-demand mapping system based on an ontology*”. The following objectives were defined:

1. To model the process of generalisation using an ontology.
2. To devise a method for automatically selecting the appropriate algorithms for mapping geographic features at multiple scales using the ontology.
3. To devise a method for generating parameter values for the selected algorithms parameters based on the current conditions in the mapped data.
4. To develop an on-demand mapping prototype for road accidents and roads that will test the above methods.

The objectives were expressed as a number of research questions:

- *How can the process of cartographic generalisation be captured in an ontology?*
 - *What are the essential characteristics of generalisation operators?*
 - *What are the essential characteristics of generalisation algorithms?*

The process of generalisation was captured using the generalisation model of McMaster and Shea (1992). The ontology captured the concepts of their model but the sequencing of

the tasks required to generalise geographic features was expressed in the control knowledge of the prototype. The characteristics of both generalisation operators and algorithms were defined (sections 5.4.2 and 5.4.3). However, in both cases the list of characteristics needs to be refined and more of the characteristics need to be represented in the ontology. This should be done with the involvement of domain experts using the methods developed in this research.

- *How can knowledge of the geographic data (accidents and roads) be described in the ontology in such a way that it can be used to guide the process of on-demand mapping?*
 - *What are the essential characteristics of a geographic feature type that effect how features of that type are generalised?*

The characteristics of the two feature types was expressed in the ontology; the road segments, for example, were classed as a *network*, which influenced how they were generalised. However, more of the characteristics of geographic features need to be represented to make full use of the ontological approach. Again this should be done with the involvement of domain experts using the hierarchy of feature types developed here as a starting template (section 5.4.1).

- *Can we automatically determine the conditions under which the data should be generalised?*

Measures for determining a single geometric condition, *congestion*, were developed; although techniques for assessing *imperceptibility* were also appraised (section 7.3.3). The application of different measure algorithms at different scales needs to be investigated further. For example, the application of different measure algorithms to the features in Figure 5.2 and those in Figure 6.34 would be beneficial.

- *Can the ontology be reasoned with, using inference, to automatically select the generalisation operators and algorithms that will resolve particular conditions in the mapped data?*
 - *Can this be done by using semantics, expressed as declarative knowledge, rather than procedural knowledge?*

The ontology was used to select generalisation operators and algorithms although the emphasis needs to be moved from reasoning about classes using OWL queries to inference

based on classifying individuals (section 7.2.2). There is not always a clear distinction between procedural knowledge and the expression of the semantics of generalisation using declarative knowledge but in general it was the latter that was represented in the ontology.

- *Once an algorithm has been selected can values for its parameters be automatically generated?*

The DoG concept was developed to allow for the automatic parameterisation of algorithms. This was successful but required the development of a number of algorithms that used the DoG value as a parameter. A technique is required where the DoG can be translated into parameter values for existing algorithms. The development of such a technique could be supported by the method developed here for assessing algorithm parameters (Table 5.5).

- *Can the geographic features data and ontology be combined in an on-demand mapping system?*

A prototype was developed for mapping accidents and road segments at multiple scales, with the limitations discussed in sections 6.9 and 7.2.

In summary, all of the objectives were met sufficiently to ensure that the aim of developing an on-demand mapping system based on an ontology was reached, although only for a specific use case. The previous chapter discussed how the ontology might be extended to make it applicable to other use cases (section 7.3). The following section will look at roles for a generalisation ontology beyond the implementation of the McMaster and Shea model.

8.2 Further work: beyond the McMaster and Shea Model

This section looks at how a generalisation ontology could be used for applications other than implementing the McMaster and Shea model in a standalone on-demand mapping system.

8.2.1 Supporting existing generalisation systems

The current design of the on-demand mapping system involves cycling through each symptom of each feature collection and attempting to resolve it. It can be argued that this linear approach is too simple and generalisation is a more complex problem. It may be that there are conditions which are best resolved by a sequence of operators, whereas the ontology, currently only suggests atomic solutions.

Agent-based systems have been employed by a number of NMAs to resolve this problem (section 2.5). However, as discussed earlier, such systems are primed to map a fixed set of

feature types at particular scales. The knowledge that these system employ is also embedded within; if we wish to map an unfamiliar feature type then this local knowledge has to be updated (Taillandier & Taillandier, 2012).

Agent-based systems could be made more flexible by providing a shared, formalised knowledge base founded on an ontology. The amount of knowledge required just to map road accidents and the underlying road network is considerable, as was demonstrated above. It is advantageous to share this knowledge and the knowledge required to map other feature types at multiple scales. A method where existing systems could interact with an ontology would need to be developed.

8.2.2 Supporting web services

There has been a lot of research on geospatial web services in general (Fitzner et al., 2011; Friis-Christensen et al., 2007; Lemmens et al., 2007) and generalisation web services in particular (Regnauld et al., 2014; Gould, 2012; Foerster et al., 2008; Burghardt et al., 2005). The standard for implementing geospatial web services is the OGC's Web Processing Service (WPS) protocol (Open Geospatial Consortium, 2010). However, the protocol does not provide for *semantic interoperability* (Janowicz et al., 2010); there is no method of adding machine readable descriptions to a service. It is possible to determine from the input and output parameters that the algorithm might input point data and output polygon data but no semantics can be determined. Human-readable service descriptions can be added but there is no way that a computer can reason about why a particular processing service should be selected.

One possible solution is the *Semantic Enablement Layer* (Janowicz et al., 2010), which *injects* semantics into both data and processing services using ontologies (Figure 8.1). In the use case the same ontology was used to define spatial data and generalisation concepts such as the operator. In this example, separate ontologies for data and generalisation are depicted. A *Web Ontology Service* is used to manage the ontologies, including updating, and a *Web Reasoning Service* is then used to reason with the ontologies; to match a particular processing (or generalisation) service to a feature collection, for example.

The WEBGEN portal (Dresden University of Technology, 2014; Neun et al., 2013) offers a number of WPS-based cartographic generalisation web services, including building displacement, building simplification, and line smoothing. These services are based on the

WPS protocol and could be employed as a use case for the Semantic Enablement Layer concept.

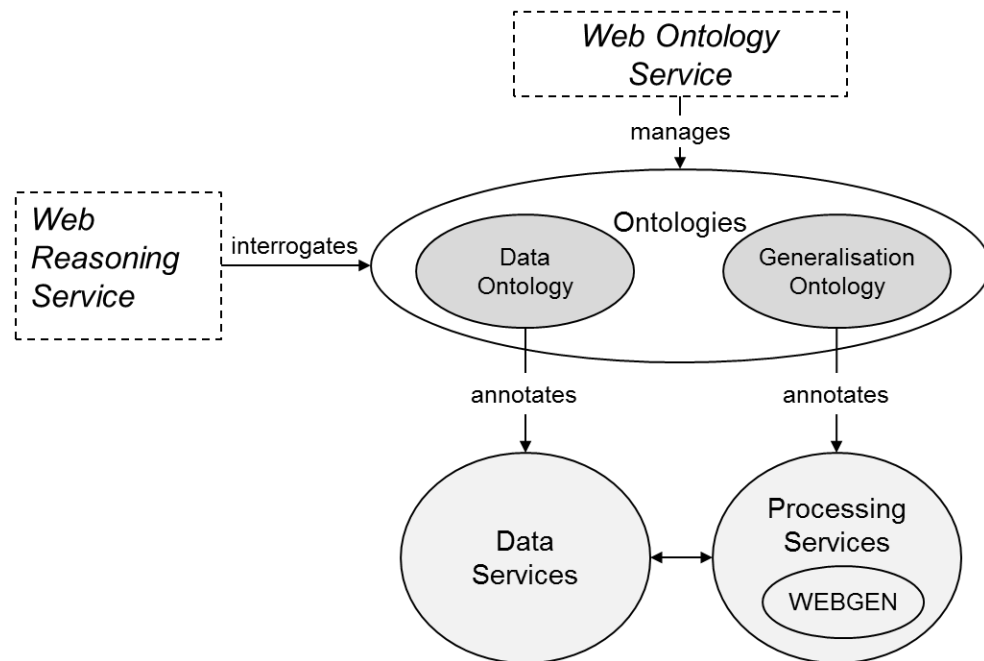


Figure 8.1 Semantic injection of generalisation web services (based on Janowicz et al., 2010)

Both of these suggested expansions of the ontological approach (integrating with existing systems and supporting web services) will require collaborative working. There are still a number of unresolved matters regarding the design of the ontology and they can be best resolved by involving domain experts. The open source *webProtégé* ontology editor (Tudorache et al., 2013) allows for the collaborative development of ontologies and can be downloaded and installed on a server or used via the Stanford University hosted implementation. *webProtégé* supports the OWL 2 standard for ontologies (Grau et al., 2008). Collaboration is supported by features such as change tracking, and a discussion forum for each class defined in the ontology (Figure 8.2). Indeed it might be possible to use *webProtégé* for the Web Ontology Service component of the Semantic Enablement Layer (Figure 8.1).

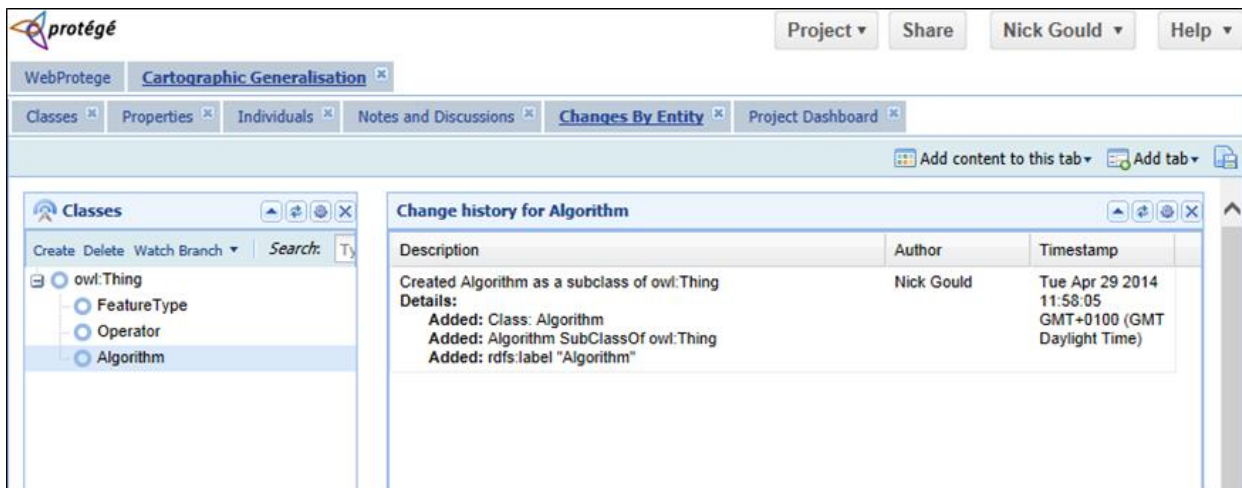


Figure 8.2 The webProtégé interface

8.3 Major achievements

The major achievements can be divided between those relating to on-demand mapping and those relating to the use of ontologies for generalisation. This thesis has

- **Defined the requirements for on-demand mapping** including the need to respect relationships between thematic features and topographic features when generalising the latter. This was demonstrated by the development of an algorithm for pruning a road network with respect to point features.
- **Developed a method for the automatic calculation of algorithm parameters** based on the DegreeOfGeneralisation concept. The output of a set of measure algorithms was used to derive the DoG which was then used as a parameter to a number of generalisation algorithms.
- **Demonstrated the need to formalise and make explicit generalisation knowledge.** Generalisation is a complex process and automatic generalisation of familiar features at a fixed scale is difficult task. On-demand mapping, which requires automatic generalisation of any feature at arbitrary scale, involves another level of difficulty. Formalising this knowledge, making it explicit, and allowing it to be shared by generalisation systems will ease that difficulty.
- **Developed a method for producing a consistent definition of generalisation operators.** Many generalisation operator classifications or taxonomies have been developed over the past 50 years. There has been much disagreement over the

meaning of terms and the scope of operators. A technique was developed that involved producing characteristic *signatures* of operators that can ensure consistency when defining an operator's characteristics. Although the technique requires further development, it could be used in the future to harmonise generalisation operator definitions.

- **Developed a methodology specifically for the design of *task* ontologies.** There are a large number of methodologies for designing *domain* ontologies but none was discovered that was aimed at designing a *task* ontology. In addition, many of the methodologies were designed for a specific domain. A hybrid methodology was developed, based on a number of existing methodologies, specifically to develop a task ontology.
- **Modelled the complete process of generalisation in an ontology for the first time.** The use of ontologies in generalisation has been proposed before but their use has been restricted to specific aspects of generalisation such as classifying features or supporting a particular algorithm. The ontology developed here attempted to encapsulate the complete process of generalisation and was evaluated by developing an on-demand mapping prototype. Many ontologies are designed, some are built, but few are applied to specific tasks. The ontology developed here requires further development, but the methods developed in this thesis that will aid the development of a comprehensive generalisation ontology.

8.4 Final words

The introductory chapter discussed the recent maps generated automatically, using a number of pre-defined workflows, by the Netherlands' Kadaster (section 1.3). In particular, their use of a *road* network pruning algorithm to prune the *hydrographic* network was mentioned as just one of the problems in attempting to completely automate the process. It might be concluded that such counterintuitive decision-making might make the aim of on-demand mapping futile. Such a decision is certainly beyond the current version of the ontology but not beyond a future version. It is a case of defining the characteristics of the algorithm that make it suitable for this particular hydrographic network and the characteristics of the network that make it appropriate to use this algorithm. If these characteristics are described with sufficient detail then the link between data and algorithm can be inferred.

The initial aim of the research was to develop a method for completely automating the production of a map with arbitrary content at an arbitrary scale. This ambition was tempered by experience. However, if the aim is not to reproduce the detailed multi-purpose maps produced by National Mapping Agencies but instead to allow the user to map a set of thematic features, plus one or two topographic feature types to provide context, then on-demand mapping does not appear to be an unattainable task. In addition, an on-demand mapping system that allows the user some control over the output is desirable.

The ambition of generalisation was defined in Chapter 1 as allowing us the ability to “see different patterns, different relationships, between different entities” (Mackaness et al., 2014). The explicit formalisation of both the process of generalisation and the knowledge of these entities in an ontology will aid that ambition by shifting the focus of generalisation from geometry to semantics. However, the *Why*, *When* and *How* model of generalisation of McMaster and Shea needs to be updated. The ambition gives reasons *why* we generalise. We generalise *when* comprehension is threatened. Finally, *how* we generalise is by respecting the semantic relationships between these entities.

References

- Akerkar, R. and Sajja, P. (2010) *Knowledge-based systems*. Jones & Bartlett Publishers.
- Antoniou, G. and van Harmelen, F. (2009) Web Ontology Language: OWL. In Staab, S. and Studer, R. (eds.) *Handbook on Ontologies*. Berlin/Heidelberg: Springer-Verlag.
- Armstrong, M. P. (1991) Knowledge classification and organization. In Buttenfield, B. and McMaster, R. B. (eds.) *Map Generalization: Making rules for knowledge representation*. New York: Longman Scientific & Technical.
- Baglioni, M., Masserotti, M., Renso, C. and Spinsanti, L. (2007) Building Geospatial Ontologies from Geographical Databases. In Fonseca, F., Rodríguez, M. A. and Levashkin, S. (eds.) *GeoSpatial Semantics*. Vol. 4853. Berlin/Heidelberg: Springer, pp. 195-209.
- Balley, S. and Regnauld, N. (2011a) Collaboration for Better On-demand Mapping. In *14th Workshop of the ICA commission on Generalisation and Multiple Representation*. Paris, France, 30th June - 1st July 2011.
- Balley, S. and Regnauld, N. (2011b) Models and standards for on-demand mapping. In *25th International Cartographic Conference*. Paris, 3-8 July 2011.
- Balley, S., Jaara, K. and Regnauld, N. (2012) Towards a Prototype for Deriving Custom Maps from Multisource Data. In *15th Workshop of the ICA commission on Generalisation and Multiple Representation*. Istanbul, Turkey, 13th-15th September 2012.
- Balley, S., Baella, B., Christophe, S., Pla, M., Regnauld, N. and Stoter, J. (2014) Map Specifications and User Requirements. In Burghardt, D., Duchene, C. and Mackaness, W. A. (eds.) *Abstracting Geographic Information in a Data Rich World*. Heidelberg: Springer, pp. 17-52.
- Bashar, M. A., Islam, M., Chowdhury, M. A., Sajjad, M. P. and Ahmed, M. T. (2012) Concept for a Web Map Implementation with Faster Query Response. *Journal of Information Engineering and Applications*, 2(1).
- Batty, M., Hudson-Smith, A., Milton, R. and Crooks, A. (2010) Map mashups, Web 2.0 and the GIS revolution. *Annals of GIS*, 16(1) pp. 1-13.
- Beard, M. K. (1991) Constraints on rule formation. In Buttenfield, B. and McMaster, R. B. (eds.) *Map Generalization: Making rules for knowledge representation*. New York: Longman Scientific & Technical New York.
- Benjamins, V. R., Fensel, D. and Gomez Perez, A. (1998) Knowledge Management through Ontologies. In *Second International Conference on Practical Aspects Knowledge Management*. Basel, Switzerland, 29th-30th October 1998.
- Benz, S. A. and Weibel, R. (2013) Road network selection using an extended stroke-mesh combination algorithm. In *16th Workshop of the ICA commission on Generalisation and Multiple Representation*. Dresden, 23rd–24th August 2013.
- Bereuter, P. and Weibel, R. (2012) Algorithms for On-the-Fly Generalization of Point Data Using Quadrees. In *17th Autocarto conference*. Columbus, Ohio.
- Bernier, E. and Bedard, Y. (2007) A Data Warehouse Strategy for on-Demand Multiscale Mapping. In Mackaness, W. A., Ruas, A. and Sarjakoski, L. T. (eds.) *Generalisation of Geographic Information*. Amsterdam: Elsevier Science B.V., pp. 177-198.
- Bertin, J. (1983) *Semiology of Graphics*. ESRI Press.

- Bobzien, M., Burghardt, D., Petzold, I., Neun, M. and Weibel, R. (2008) Multi-representation databases with explicitly modeled horizontal, vertical, and update relations. *Cartography and Geographic Information Science*, 35(1).
- Borst, W. (1997) *Construction of Engineering Ontologies*. PhD. University of Twente.
- Brassel, K. E. and Weibel, R. (1988) A review and conceptual framework of automated map generalization. *International Journal of Geographical Information Systems*, 2(3) pp. 229-244.
- Brauner, J., Foerster, T., Schaeffer, B. and Baranski, B. (2009) Towards a Research Agenda for Geoprocessing Services. In *12th AGILE International Conference on Geographic Information Science*. Leibniz Universität Hannover, Germany.
- Bucher, B., Falquet, G., Clementini, E. and Sester, M. (2012) Towards a typology of spatial relations and properties for urban applications. In *Usage, Usability, and Utility of 3D City Models*. Nantes, France, 29th-31st October 2012.
- Burghardt, D. (2005) Controlled Line Smoothing by Snakes. *Geoinformatica*, 9(3) pp. 237-252.
- Burghardt, D., Neun, M. and Weibel, R. (2005) Generalization Services on the Web - Classification and an Initial Prototype Implementation. *Cartography and Geographic Information Science*, 32(4) pp. 257-268.
- Burghardt, D., Schmid, S. and Stoter, J. (2007) Investigations on Cartographic Constraint Formalisation. In *10th Workshop of the ICA commission on Generalisation and Multiple Representation*. Moscow, Russia, 2nd-3rd August 2007.
- Buttenfield, B. and McMaster, R. B. (1991) *Map Generalization: Making rules for knowledge representation*. New York: Longman Scientific & Technical New York.
- Bylander, T. and Chandrasekaran, B. (1987) Generic tasks for knowledge-based reasoning: the “right” level of abstraction for knowledge acquisition. *International Journal of Man-Machine Studies*, 26(2), pp. 231-243.
- Carral, D., Scheider, S., Janowicz, K., Vardeman, C., Krisnadhi, A. and Hitzler, P. (2013) An Ontology Design Pattern for Cartographic Map Scaling. In Cimiano, P., Corcho, O., Presutti, V., Hollink, L. and Rudolph, S. (eds.) *The Semantic Web: Semantics and Big Data*. Vol. 7882. Springer: Berlin/Heidelberg, pp. 76-93.
- Cecconi, A. (2003) *Integration of cartographic generalization and multi-scale databases for enhanced web mapping*. PhD. University of Zurich.
- Cecconi, A., Weibel, R. and Barrault, M. (2002) Improvement of Automated Generalization for On-demand Mapping by Multiscale Databases. In *Geospatial Theory, Processing and Applications, ISPRS Commission IV, Symposium 2002*. Ottawa, Canada.
- Chang, H. and McMaster, R. (1993) Interface Design and Knowledge Acquisition for Cartographic Generalization. In *AutoCarto 11*. Minneapolis, 30th October - 1st November, 1993.
- Chaudhry, O. Z. and Mackaness, W. A. (2005) Rural and Urban Road Network Generalisation Deriving 1:250,000 from OS MasterMap. In *22nd International Cartographic Conference*. A Coruña, Spain, 11th-16th July 2005.
- Chen, J., Hu, Y., Li, Z., Zhao, R. and Meng, L. (2009) Selective omission of road features based on mesh density for automatic map generalization. *International Journal of Geographical Information Science*, 23(8) pp. 1013-1032.

- Chen, T.-Y. (2008) Knowledge sharing in virtual enterprises via an ontology-based access control approach. *Computers in Industry*, 59(5) pp. 502-519.
- Clancey, W. J. (1985) Heuristic classification. *Artificial Intelligence*, 27(3), pp. 289-350.
- Codd, E. F. (1970) A relational model of data for large shared data banks. *Communications of the ACM*, 13(6) pp. 377-387.
- Corcoran, P., Mooney, P. and Bertolotto, M. (2011) Utilizing geometric coherence in the computation of map transformations. *Computers and Geosciences*, 47 pp. 151–159.
- Corcoran, P., Mooney, P. and Bertolotto, M. (2012) Spatial Relations Using High Level Concepts. *ISPRS International Journal of Geo-Information*, 1(3) pp. 333-350.
- Damen, J., van Kreveld, M. and Spaan, B. (2008) High Quality Building Generalization by Extending the Morphological Operators. In *11th Workshop of the ICA commission on Generalisation and Multiple Representation*. Montpellier, France, 20th-21st June 2008.
- Davis, R., Shrobe, H. and Szolovits, P. (1993) What is a knowledge representation? *AI magazine*, 14(1).
- De Nicola, A., Missikoff, M. and Navigli, R. (2009) A software engineering approach to ontology building. *Information Systems*, 34(2) pp. 258-275.
- Denaux, R., Dolbear, C., Hart, G., Dimitrova, V. and Cohn, A. G. (2011) Supporting domain experts to construct conceptual ontologies: A holistic approach. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(2) pp. 113-127.
- Douglas, D. H. and Peucker, T. K. (1973) Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2) pp. 112-122.
- Dresden University of Technology. (2014) *WebGen-WPS Portal*. [Online] [Accessed on 12th June 2014] http://kartographie.geo.tu-dresden.de/webgen_docs/
- Dunkars, M. (2004) Automated Generalisation In A Multiple Representation Database. In *12th Int. Conf. on Geoinformatics - Geospatial Information Research: Bridging the Pacific and Atlantic*. University of Gävle, Sweden.
- Dutton, G. and Edwardes, A. (2006) Ontological Modeling of Geographical Relationships for Map Generalization. In *Workshop of the ICA Commission on Map Generalization and Multiple Representation*. Portland, USA, 25th June 2006.
- Eckert, M. (1908) On The Nature Of Maps And Map Logic. *Bulletin of the American Geographical Society*, 40(6) pp. 344-351.
- Egenhofer, M. J. and Franzosa, R. D. (1991) Point-set topological spatial relations. *International Journal of Geographical Information Systems*, 5(2) pp. 161-174.
- ESRI. (2011a) *Aggregate Polygons*. [Online] [Accessed on 12th June 2014] http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#/Aggregate_Polygons/007000000000s000000/
- ESRI. (2011b) *arcGIS Aggregate Points*. [Online] [Accessed on 12th June 2014] <http://resources.arcgis.com/en/help/main/10.1/index.html#/007000000002s0000000>

- ESRI. (2012) *Thin Road Network (Cartography)*. [Online] [Accessed on 12th June 2014]
http://resources.arcgis.com/en/help/main/10.1/index.html#/Thin_Road_Network/007000000014000000/
- ESRI. (2014) *GIS Dictionary*. [Online] [Accessed on 12th June 2014]
<http://support.esri.com/en/knowledgebase/Gisdictionary>
- Ester, M., Kriegel, H.-P., J., S. and X., X. (1996) A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *2nd international Conference on Knowledge Discovery and Data Mining* Portland, Oregon, USA.
- Fan, H. and Meng, L. (2010) A generic approach for simplification of building ground plan. In *13th Workshop of the ICA commission on Generalisation and Multiple Representation*. Zurich, Switzerland, 12th-13th September 2010.
- Fernández-López, M. and Gómez-Pérez, A. (2002) Overview and analysis of methodologies for building ontologies. *Knowledge Engineering Review*, 17(2) pp. 129-156.
- Fernández-López, M., Gómez-Pérez, A. and Juristo, N. (1997) *Methontology: from ontological art towards ontological engineering*. Stanford University, 24-26 March 1997. American Association for Artificial Intelligence.
- Fisher, P. F. and Mackaness, W. A. (1987) Are cartographic expert systems possible? In *Proceedings AutoCarto*. Vol. 8, pp. 530-534.
- Fitzner, D., Hoffmann, J. and Klien, E. (2011) Functional description of geoprocessing services as conjunctive datalog queries. *Geoinformatica*, 15(1) pp. 191-221.
- Foerster, T., Stoter, J. and Köbben, B. (2007a) Towards a formal classification of generalization operators. In *International Cartographic Conference 2007*. Moscow, 4th-10th August, 2007.
- Foerster, T., Stoter, J. and van Oosterom, P. (2012) On-demand base maps on the web generalized according to user profiles. *International Journal of Geographical Information Science*, 26(1), pp. 99-121.
- Foerster, T., stoter, J., Köbben, B. and van Oosterom, P. (2007b) A Generic Approach to Simplification of Geodata for Mobile Applications. In *10th AGILE International Conference on Geographic Information Science*. Aalborg University, Denmark.
- Foerster, T., Burghardt, D., Neun, M., Regnaud, N., Swan, J. and Weibel, R. (2008) Towards an Interoperable Web Generalisation Services Framework - Current work in progress. In *11th Workshop of the ICA commission on Generalisation and Multiple Representation*. Montpellier, France, 20th-21st June 2008.
- Fonseca, F., Egenhofer, M., Davis, C. and Câmara, G. (2002) Semantic Granularity in Ontology-Driven Geographic Information Systems. *Annals of Mathematics and Artificial Intelligence*, 36(1-2) pp. 121-151.
- Fox, J. (2011) Formalizing knowledge and expertise: where have we been and where are we going? *The Knowledge Engineering Review*, 26(Special Issue 01) pp. 5-10.
- Friis-Christensen, A., Ostländer, N., Lutz, M. and Bernard, L. (2007) Designing Service Architectures for Distributed Geoprocessing: Challenges and Future Directions. *Transactions in GIS*, 11(6) pp. 799-818.
- Gangemi, A. and Presutti, V. (2009) Ontology Design Patterns. In Staab, S. and Studer, R. (eds.) *Handbook on Ontologies*. Berlin/Heidelberg: Springer-Verlag.

- García-Sánchez, F., Martínez-Béjar, R., Contreras, L., Fernández-Breis, J. T. and Castellanos-Nieves, D. (2006) An ontology-based intelligent system for recruitment. *Expert Systems with Applications*, 31(2) pp. 248-263.
- Genesereth, M. R. and Nilsson, N. J. (1998) *Logical foundations of artificial intelligence*. Palo Alto, California: Morgan Kaufmann Publishers.
- GeoTools. (2014) *GeoTools*. [Online] [Accessed on 12th June 2014] <http://www.geotools.org/>
- Gómez Pérez, A. and Benjamins, R. (1999) Overview of Knowledge Sharing and Reuse Components: Ontologies and Problem-Solving Methods. In *16th International Joint Conference on Artificial Intelligence*. Stockholm, Sweden, 2nd August 1999.
- Gould, N. (2012) Semantic description of generalisation web services for on-demand mapping. In *1st AGILE PhD School*. Wernigerode, Germany, 13-14th March 2012. Bernard, L. and Pundt, H.
- Gould, N. and Chaudhry, O. Z. (2012) Generation and validation of workflows for on-demand mapping. In *Geoprocessing 2012*. Valencia, Spain, 30th January - 4th February 2012.
- Grabler, F., Agrawala, M., Sumner, R. W. and Pauly, M. (2008) Automatic generation of tourist maps. In *ACM SIGGRAPH 2008 papers*. Los Angeles, California.
- Grau, B. C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P. and Sattler, U. (2008) OWL 2: The next step for OWL. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4) pp. 309-322.
- Gruber, T. (1993) A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2) pp. 199-220.
- Gruninger, M., Bodenreider, O., Olken, F., Obrst, L. and Yim, P. (2008) Ontology Summit 2007 – Ontology, taxonomy, folksonomy: Understanding the distinctions. *Applied Ontology*, 3(3) pp. 191-200.
- Grüninger, M. and Fox, M. (1995) Methodology for the Design and Evaluation of Ontologies. In *IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing*. Montreal, 19-20 August 1995.
- Guarino, N. and Welty, C. A. (2002) Evaluating ontological decisions with OntoClean. *Communications of the ACM*, 45(2) pp. 61-65.
- Guarino, N., Oberle, D. and Staab, S. (2009) What is an Ontology? In Staab, S. and Studer, R. (eds.) *Handbook on Ontologies*. 2nd ed., Berlin/Heidelberg: Springer-Verlag.
- Guercke, R. and Sester, M. (2011) Building Footprint Simplification Based on Hough Transform and Least Squares Adjustment. In *14th Workshop of the ICA commission on Generalisation and Multiple Representation*. Paris, France, 30th June - 1st July 2011.
- Hampe, M., Anders, K. and Sester, M. (2003) *MRDB Applications for data revision and realtime generalisation*.
- Harrie, L. and Weibel, R. (2007) Modelling the Overall Process of Generalisation. In Mackaness, W. A., Ruas, A. and Sarjakoski, L. (eds.) *Generalisation of Geographic Information*. Amsterdam: Elsevier Science B.V., pp. 67-87.
- Harrie, L. and Stigmar, H. (2010) An evaluation of measures for quantifying map information. *ISPRS Journal Of Photogrammetry And Remote Sensing*, 65(3) pp. 266-274.
- Hart, G. and Dolbear, C. (2013) *Linked Data: A Geographic Perspective*. Boca Raton, USA: CRC Press.

- Horridge, M. and Patel-Schneider, P. F. (2008) Manchester syntax for OWL 1.1. In *International Workshop OWL: Experiences and Directions (OWLED08)*. Washington, DC, USA.
- Horridge, M. and Patel-Schneider, P. (2009) *OWL 2 Web Ontology Language Manchester Syntax*. [Online] [Accessed on 4th June 2014] www.w3.org/TR/owl2-manchester-syntax/
- Horridge, M. and Bechhofer, S. (2011) The OWL API: A Java API for OWL ontologies. *Semantic Web*, 2(1) pp. 11-21.
- Horrocks, I. (2013) What Are Ontologies Good For? In Küppers, B.-O., Hahn, U. and Artmann, S. (eds.) *Evolution of Semantic Systems*. Vol. Springer. Berlin Heidelberg, pp. 175-188.
- Horrocks, I., Patel-Schneider, P., Boley, H., Tabet, S., Grosof, B. and Dean, M. (2004) *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. [Online] [Accessed on 12th June 2014] <http://www.w3.org/Submission/SWRL/>
- Hubert, F. and Ruas, A. (2003) A method based on samples to capture user needs for generalisation. In *Fifth ICA workshop on progress in automated map generalisation*. Paris, France.
- Imhof, E. (1982) *Cartographic relief presentation*. Berlin: Walter de Gruyter.
- International Cartographic Association (1973) *Multilingual Dictionary of Technical Terms in Cartography*. Franz Steiner Verlag.
- Iosifescu-Enescu, I. and Hurni, L. (2007) Towards cartographic ontologies or “how computers learn cartography”. In *23rd International Cartographic Conference*. Moscow, Russia, 4th-9th August 2007.
- Iqbal, R., Murad, M. A. A., Mustapha, A. and Sharef, N. M. (2013) An Analysis of Ontology Engineering Methodologies: A Literature Review. *Research Journal of Applied Sciences, Engineering and Technology*, 16(6).
- Jaara, K., Duchene, C. and Ruas, A. (2012) A Model for Preserving the Consistency between Topographic and Thematic Layers throughout Data Migration. In *Proceedings of the Short Papers of the SDH2012*. Bonn, Germany, 21st-24th August 2012.
- Janowicz, K., Scheider, S., Pehle, T. and Hart, G. (2012) Geospatial semantics and linked spatiotemporal data – Past, present, and future. *Semantic Web*, 3(4) pp. 321-332.
- Janowicz, K., Schade, S., Broring, A., Kessler, C., Maué, P. and Stasch, C. (2010) Semantic Enablement for Spatial Data Infrastructures. *Transactions in GIS*, 14(2) p. 111.
- Janssen, M., Charalabidis, Y. and Zuiderwijk, A. (2012) Benefits, Adoption Barriers and Myths of Open Data and Open Government. *Information Systems Management*, 29(4) pp. 258-268.
- Jenks, G. F. (1989) Geographic Logic In Line Generalization. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 26(1) pp. 27-42.
- Jiang, B. and Claramunt, C. (2004) A Structural Approach to the Model Generalization of an Urban Street Network. *Geoinformatica*, 8(2) pp. 157-171.
- Jiang, B. and Harrie, L. (2004) Selection of Streets from a Network Using Self-Organizing Maps. *Transactions in GIS*, 8(3) pp. 335-350.
- Jiang, B. and Liu, C. (2009) Street-based topological representations and analyses for predicting traffic flow in GIS. *International Journal of Geographical Information Science*, 23(9), 2012/07/05, pp. 1119-1137.
- João, E. M. (1998) *Causes and consequences of map generalisation*. London: Taylor and Francis.

- Jung, C.-T., Sun, C.-H. and Yuan, M. (2013) An ontology-enabled framework for a geospatial problem-solving environment. *Computers, Environment and Urban Systems*, 38(0) pp. 45-57.
- Kavouras, M. and Kokla, M. (2008) *Theories of geographic concepts : ontological approaches to semantic integration*. Boca Raton: CRC Press.
- Kilpeläinen, T. (2000) Knowledge Acquisition for Generalization Rules. *Cartography and Geographic Information Science*, 27(1).
- Kingston, J. K. C. (1998) Designing knowledge based systems: the CommonKADS design model. *Knowledge-Based Systems*, 11(5–6) pp. 311-319.
- Kontopoulos, E., Vrakas, D., Kokkoras, F., Bassiliades, N. and Vlahavas, I. (2008) An ontology-based planning system for e-course generation. *Expert Systems with Applications*, 35(1–2) pp. 398-406.
- Kopf, J., Agrawala, M., Barger, D., Salesin, D. and Cohen, M. (2010) Automatic generation of destination maps. *ACM Transactions on Graphics*, 29(6) pp. 1-12.
- Kulik, L., Duckham, M. and Egenhofer, M. (2005) Ontology-driven map generalization. *Journal of Visual Languages & Computing*, 16(3) pp. 245-267.
- Lagrange, F., Landras, B. and Mustiere, S. (2000) Machine learning techniques for determining parameters of cartographic generalisation algorithms. *International Archives Of Photogrammetry And Remote Sensing*, 33.
- Lamy, S., Ruas, A., Demazeau, Y., Jackson, M., Mackaness, W. A. and Weibel, R. (1999) *The application of agents in automated map generalisation*. Ottawa, 14th-21st August 1999.
- Laurini, R. (2012) Importance of spatial relationships for geographic ontologies. In *Proceedings of the Seventh International Conference on Informatics and Urban and Regional Planning*. Cagliari, Italy, 10th-12th May 2012. Campagna, M., De Montis, A., Isola, F., Lai, S., Pira, C. and Zoppi, C. pp. 122-134.
- Lecordix, F. and Lemarie, C. (2007) Managing Generalisation Updates in IGN Map Production. In Mackaness, W. A., Anne, R. and Sarjakoski, L. T. (eds.) *Generalisation of Geographic Information*. Amsterdam: Elsevier Science B.V., pp. 285-300.
- Lemmens, R., De By, R., Gould, M., Wytzisk, A., Granell, C. and Van Oosterom, P. (2007) Enhancing Geo-Service Chaining through Deep Service Descriptions. *Transactions in GIS*, 11(6) pp. 849-871.
- Li, Z. (2006) *Algorithmic foundation of multi-scale spatial representation*. 1st ed.: CRC Press.
- Li, Z. (2007) Digital map generalization at the age of enlightenment: A review of the first forty years. *The Cartographic Journal*, 44(1) pp. 80-93.
- Li, Z. and Zhou, Q. (2012) Integration of linear and areal hierarchies for continuous multi-scale representation of road networks. *International Journal of Geographical Information Science*, 26(5) pp. 855-880.
- Liu, X., Zhan, F. B. and Ai, T. (2010) Road selection based on Voronoi diagrams and strokes in map generalization. *International Journal of Applied Earth Observation and Geoinformation*, 12(Supplement 2) pp. S194-S202.
- Longley, P., Goodchild, M., Maguire, D. and Rhind, D. (2005) *Geographic Information Systems and Science*. John Wiley & Sons.

- Luscher, P., Weibel, R. and Mackaness, W. A. (2008) Where is the Terraced House? On the Use of Ontologies for Recognition of Urban Concepts in Cartographic Databases. *In* Ruas, A. and Gold, C. (eds.) *Headway in Spatial Data Handling*. Springer Berlin Heidelberg, pp. 449-466.
- Ma, X., Carranza, E. J. M., Wu, C. and van der Meer, F. D. (2012) Ontology-aided annotation, visualization, and generalization of geological time-scale information from online geological map services. *Computers and Geosciences*, 40 pp. 107-119.
- Mackaness, W., A., Burghardt, D. and Duchene, C. (2014) Map Generalisation: Fundamental to the Modelling and Understanding of Geographic Space. *In* Burghardt, D., Duchene, C. and Mackaness, W. A. (eds.) *Abstracting Geographic Information in a Data Rich World*. Heidelberg: Springer, pp. 299-328.
- Mackaness, W. A. (2007) Understanding Geographic Space. *In* Mackaness, W. A., Ruas, A. and Sarjakoski, L. T. (eds.) *Generalisation of Geographic Information*. Amsterdam: Elsevier Science B.V., pp. 1-10.
- Mackaness, W. A. and Ruas, A. (2007) Evaluation in the Map Generalisation Process. *In* Mackaness, W., A. , Ruas, A. and Sarjakoski, L. T. (eds.) *Generalisation of Geographic Information*. Amsterdam: Elsevier Science B.V., pp. 89-111.
- Mackaness, W. A. and Reimer, A. (2014) Generalisation in the Context of Schematic Maps. *In* Burghardt, D., Duchene, C. and Mackaness, W. A. (eds.) *Abstracting Geographic Information in a Data Rich World*. Heidelberg: Springer, pp. 299-328.
- MacQueen, J. B. (1966) Some methods for classification and analysis of multivariate observations. *In* *Fifth Berkeley Symposium on Mathematical Statistics and Probability*. University of California, Berkeley.
- Man, L., Xiao-Yong, D. and Shan, W. (2005) Learning ontology from relational database. *In* *Proceedings of 2005 International Conference on Machine Learning and Cybernetics*. Vol. 6. Guangzhou, China, 18th – 21st August 2005.
- Martinez-Cruz, C., Blanco, I. and Vila, M. A. (2012) Ontologies versus relational databases: are they so different? A comparison. *Artificial Intelligence Review*, 38(4) pp. 271-290.
- Matsuo, J., Kitayama, D., Lee, R. and Sumiya, K. (2011) Modified map search engine: geographical features extraction for ranking of modified maps. *In* *Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication*. Seoul, Korea.
- McMaster, R. (1989) The Integration Of Simplification And Smoothing Algorithms In Line Generalization. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 26(1) pp. 101-121.
- McMaster, R. (1991) Conceptual frameworks for geographical knowledge. *In* Buttenfield, B. and McMaster, R. B. (eds.) *Map Generalization: Making rules for knowledge representation*. New York: Longman Scientific & Technical New York.
- McMaster, R. B. and Shea, K. S. (1992) *Generalization in digital cartography*. Washington D.C.: Association of American Geographers.
- Meijers, M., Stoter, J. and van oosterom, P. (2012) Comparing the vario-scale approach with a discrete multi-representation based approach for automated generalisation of topographic data. *In* *15th Workshop of the ICA commission on Generalisation and Multiple Representation*. Istanbul, Turkey, 13th-15th September 2012.
- Mizoguchi, R. and Kozaki, K. (2009) Ontology Engineering Environments. *In* Staab, S. and Studer, R. (eds.) *Handbook on Ontologies*. 2nd ed., Berlin/Heidelberg: Springer-Verlag.

- Muller, J. C. (1987) Fractal and Automated Line Generalization. *The Cartographic Journal*, 24(1) pp. 27-34.
- Mustiere, S. (2005) Cartographic generalization of roads in a local and adaptive approach: A knowledge acquisition problem. *International Journal of Geographical Information Science*, 19(8-9), pp. 937-955.
- Neun, M., Burghardt, D. and Weibel, R. (2009) Automated processing for map generalization using web services. *GeoInformatica*, 13(4) pp. 425-452.
- Neun, M., Regnauld, N. and Weibel, R. (2013) WEBGEN: Web Services to Share Cartographic Generalization Tools. *In Innovative Software Development in GIS*. John Wiley & Sons, Inc., pp. 257-281.
- Niaraki, A. S. and Kim, K. (2009) Ontology based personalized route planning system using a multi-criteria decision making approach. *Expert Systems with Applications*, 36(2, Part 1) pp. 2250-2259.
- Nickerson, B. G. (1991) Knowledge engineering for generalisation. *In Buttenfield, B. and McMaster, R. B. (eds.) Map Generalization: Making rules for knowledge representation*. New York: Longman Scientific & Technical.
- Noy, N. F. and McGuinness, D. (2001) *Ontology Development 101: A Guide to Creating Your First Ontology*. Technical Report SMI-2001-0880, Stanford University.
- Nyerges, T. L. (1991) Representing geographical meaning. *In Buttenfield, B. and McMaster, R. B. (eds.) Map Generalization: Making rules for knowledge representation*. New York: Longman Scientific & Technical.
- Open Geospatial Consortium. (2010) *Web Processing Service*. [Online] [Accessed on 12th June 2014] <http://www.opengeospatial.org/standards/wps>
- Ordnance Survey. (2014a) *Meridian 2*. [Online] [Accessed on 12th June 2014] <http://www.ordnancesurvey.co.uk/business-and-government/products/meridian2.html>
- Ordnance Survey. (2014b) *Spatial relations ontology*. [Online] [Accessed on 12th June 2014] <http://data.ordnancesurvey.co.uk/ontology/spatialrelations/>
- Oxford English Dictionary (2014a) "*symptom, n.*". Oxford University Press.
- Oxford English Dictionary (2014b) "*event, n.*". Oxford University Press.
- Oxford English Dictionary (2014c) "*software, n.*". Oxford University Press.
- Qi, H. B. and Li, Z. L. (2008) An approach to building grouping based on hierarchical constraints. *In The XXI Congress of the International Society for Photogrammetry and Remote Sensing*. Beijing, China, pp. 449-454.
- Regnauld, N. (2003) Algorithms for the amalgamation of topographic data. *In Proceedings of the 21st International Cartographic Conference*, Durban.
- Regnauld, N. (2007) Evolving from automating existing map production systems to producing maps on demand automatically. *In 10th Workshop of the ICA commission on Generalisation and Multiple Representation*. Moscow, Russia, 2nd-3rd August 2007.
- Regnauld, N. and McMaster, R. B. (2007) A synoptic view of generalisation operators. *In Mackaness, W. A., Ruas, A. and Sarjakoski, L. T. (eds.) Generalisation of Geographic Information*. Amsterdam: Elsevier Science B.V., pp. 37-66.

- Regnauld, N. and Revell, P. (2007) Automatic Amalgamation of Buildings for Producing Ordnance Survey 1:50 000 Scale Maps. *The Cartographic Journal*, 44(3) pp. 239-250.
- Regnauld, N., Lessware, S., Wesson, C. and Martin, P. (2013) Deriving Products from a Multi Resolution Database using Automated Generalisation at Ordnance Survey. *In 26th International Cartographic Conference*. Dresden.
- Regnauld, N., Touya, G., Gould, N. and Foerster, T. (2014) Process Modelling, Web Services and Geoprocessing. *In Burghardt, D., Duchene, C. and Mackaness, W. A. (eds.) Abstracting Geographic Information in a Data Rich World*. Heidelberg: Springer, pp. 197-226.
- Revell, P. (2004) Building on Past Achievements: Generalising OS MasterMap Rural Buildings to 1:50,000. *In Workshop of the ICA commission on Generalisation and Multiple Representation*. Leicester, UK, 20th-21st August 2004.
- Revell, P., Regnauld, N. and Thom, S. (2005) Generalising OS MasterMap topographic Buildings and ITN road centerlines to 1:50 000 scale using a spatial hierarchy of agents, triangulation and topology. *In 22nd International Cartographic Conference*. A Coruña, Spain, 9-16 July 2005.
- Revell, P., Regnauld, N. and Bulbrooke, B. (2011) OS VectorMap District: automated generalisation, text placement and conflation in support of making public data public. *In 25th International Cartographic Conference*. Paris, 3rd-8th July 2011.
- Rich, E. and Knight, K. (1991) *Artificial intelligence*. 2nd ed., New York: McGraw-Hill.
- Richter, K.-F. and Winter, S. (2014) Introduction: What Landmarks Are, and Why They Are Important. *In Landmarks: GIScience for Intelligent Services*. Springer International Publishing, pp. 1-25.
- Rieger, M. K. and Coulson, M. R. C. (1993) Consensus or confusion: cartographers' knowledge of generalization. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 30(2) pp. 69-80.
- Roth, R., Brewer, C. and Stryker, M. (2011) A typology of operators for maintaining legible map designs at multiple scales. *Cartographic Perspectives*, 68, Winter 2011.
- Ruas, A. and Duchêne, C. (2007) A prototype generalisation system based on the multi-agent system paradigm. *In Mackaness, W. A., Anne, R. and Sarjakoski, L. T. (eds.) Generalisation of Geographic Information*. Amsterdam: Elsevier Science B.V.
- Sabo, M. N., Bedard, Y., Moulin, B. and Bernier, E. (2008) Toward Self-Generalizing Objects and On-the-Fly Map Generalization. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 43(3) pp. 155-173.
- Saripalle, R. and Demurjian, S. A. (2012) Towards a Hybrid Ontology Design and Development Life Cycle. *In International Conference on Semantic Web and Web Services (SWWS)*. Las Vegas, USA,
- Sarjakoski, L. T. (2007) Conceptual Models of Generalisation and Multiple Representation. *In Mackaness, W. A., Ruas, A. and Sarjakoski, L. T. (eds.) Generalisation of Geographic Information*. Amsterdam: Elsevier Science B.V., pp. 11-35.
- Šaša Bastinos, A. and Krisper, M. (2013) Multi-criteria decision making in ontologies. *Information Sciences*, 222 pp. 593-610.
- Scheuer, S., Haase, D. and Meyer, V. (2013) Towards a flood risk assessment ontology – Knowledge integration into a multi-criteria risk assessment approach. *Computers, Environment and Urban Systems*, 37 pp. 82-94.

- Schreiber, G., Wielinga, B., de Hoog, R., Akkermans, H. and Van de Velde, W. (1994) CommonKADS: a comprehensive methodology for KBS development. *IEEE Expert*, 9(6) pp. 28-37.
- Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., Van der Velde, W. and Wielinga, B. (2000) *Knowledge engineering and management: the CommonKADS methodology*. Cambridge, Massachusetts: MIT Press.
- Sester, M. (2005) Optimization approaches for generalization and data abstraction. *International Journal of Geographical Information Science*, 19(8-9), pp. 871-897.
- Shea, K. S. (1991) Design considerations for an artificially intelligent system. In Buttenfield, B. and McMaster, R. B. (eds.) *Map Generalization: Making rules for knowledge representation*. New York: Longman Scientific & Technical New York,
- Shearer, R., Motik, B. and Horrocks, I. (2008) HermiT: A Highly-Efficient OWL Reasoner. In *OWLED 2008: 5th International Workshop on OWL*. Karlsruhe, Germany, 26th-27th October, 2008
- Shi, X. (2011) Where are the spatial relationships in the spatial ontologies? *Proceedings of the National Academy of Sciences of the United States of America*, 108(33).
- Silverman, B. W. (1986) *Density estimation for statistics and data analysis*. Vol. 26: CRC press.
- Sommer, S. and Wade, T. (2006) *A to Z GIS: An Illustrated Dictionary of Geographic Information Systems*. Esri Press.
- Stadlhofer, B., Salhofer, P. and Durlacher, A. (2013) An Overview of Ontology Engineering Methodologies in the Context of Public Administration. In *SEMAPRO 2013, The Seventh International Conference on Advances in Semantic Processing*, pp. 36-42.
- Stanford Center for Biomedical Research. (2014) *Protégé*. [Online] [Accessed on 12th June 2014] <http://protege.stanford.edu/>
- Stanislawski, L. and Savino, S. (2011) Pruning of hydrographic networks: a comparison of two approaches. In *14th Workshop of the ICA commission on Generalisation and Multiple Representation*. Paris, France, 30th June - 1st July 2011.
- Steiniger, S., Taillandier, P. and Weibel, R. (2010) Utilising urban context recognition and machine learning to improve the generalisation of buildings. *International Journal of Geographical Information Science*, 24(2), 2010/02/01, pp. 253-282.
- Stevens, R. and Lord, P. (2012) *Managing synonymy in OWL*. Ontogenesis: [Online] [Accessed on 12th June 2014] <http://ontogenesis.knowledgeblog.org/1236>
- Stevens, R. and Sattler, U. (2013) *An object lesson in choosing between a class and an object*. Ontogenesis: [Online] [Accessed on 12th June 2014] <http://ontogenesis.knowledgeblog.org/1418>
- Stigmar, H. and Harrie, L. (2011) Evaluation of Analytical Measures of Map Legibility. *The Cartographic Journal*, 48(1) pp. 41-53.
- Stoter, J., Visser, T., van Oosterom, P., Quak, W. and Bakker, N. (2010) A semantic-rich multi-scale information model for topography. *International Journal of Geographical Information Science*, 25(5), pp. 739-763.
- Stoter, J., Post, M., van Altena, V., Nijhuis, R. and Bruns, B. (2014) Fully automated generalization of a 1:50k map from 1:10k data. *Cartography and Geographic Information Science*, 41(1) pp. 1-13.
- Studer, R., Benjamins, R. and Fensel, D. (1998) Knowledge engineering: Principles and methods. *Data and Knowledge Engineering*, 25(1-2) pp. 168-198.

- Sure, Y., Staab, S. and Studer, R. (2009) Ontology Engineering Methodology. In Staab, S. and Studer, R. (eds.) *Handbook on Ontologies*. 2nd ed., Berlin/Heidelberg: Springer-Verlag.
- Szwed, P., Kadluczka, P., Chmiel, W., Glowacz, A. and Sliwa, J. (2012) Ontology based integration and decision support in the INSIGMA route planning subsystem. In *Federated Conference on Computer Science and Information Systems (FedCSIS)*. Wroclaw, Poland, 9th - 12th September 2012 pp. 141-148.
- Taboada, M., Des, J., Jose, M. and Marin, R. (2001) Diagnosis systems in medicine with reusable knowledge components. *Intelligent Systems, IEEE*, 16(6) pp. 68-73.
- Taillandier, P. (2007) Automatic knowledge revision of a generalisation system. In *10th Workshop of the ICA commission on Generalisation and Multiple Representation*. Moscow, 2nd-3rd August 2007.
- Taillandier, P. and Taillandier, F. (2012) Multi-criteria diagnosis of control knowledge for cartographic generalisation. *European Journal of Operational Research*, 217(3) pp. 633-642.
- Taillandier, P., Duchene, C. and Drogoul, A. (2011) Automatic revision of rules used to guide the generalisation process in systems based on a trial and error strategy. *International Journal of Geographical Information Science*, 25(12) pp. 1971-1999.
- Tan, P.-N., Steinbach, M. and Kumar, V. (2006) *Introduction to Data Mining*. Addison-Wesley Longman Publishing Co., Inc.
- Thom, S. (2005) A Strategy for Collapsing OS Integrated Transport Network dual carriageways. In *22nd International Cartographic Conference*. A Coruña, Spain, 9-16 July 2005.
- Thomson, R. and Richardson, D. (1999) The 'Good Continuation' Principle of Perceptual Organization applied to the Generalization of Road Networks. In *17th International Cartographic Conference*. Ottawa.
- Tinati, R., Carr, L., Halford, S. and Pope, C. (2012) Exploring the Impact of Adopting Open Data in the UK Government. In *Digital Futures 2012*. Aberdeen, 23-25 October 2012.
- Töpfer, F. and Pillewizer, W. (1966) The Principles of Selection. *The Cartographic Journal*, 3(1) pp. 10-16.
- Torres, M., Quintero, R., Moreno-Ibarra, M., Menchaca-Mendez, R. and Guzman, G. (2011) GEONTO-MET: an approach to conceptualizing the geographic domain. *International Journal of Geographical Information Science*, 25(10) pp. 1633-1657.
- Touya, G. (2007) River network selection based on structure and pattern recognition. In *23rd International Cartographic Conference*. Moscow, 4th-9th August 2007.
- Touya, G. (2010) A Road Network Selection Process Based on Data Enrichment and Structure Detection. *Transactions in GIS*, 14(5) pp. 595-614.
- Touya, G., Duchêne, C. and Ruas, A. (2010) Collaborative Generalisation: Formalisation of Generalisation Knowledge to Orchestrate Different Cartographic Generalisation Processes. In Fabrikant, S., Reichenbacher, T., Kreveld, M. and Schlieder, C. (eds.) *Geographic Information Science*. Vol. 6292. Berlin/Heidelberg: Springer, pp. 264-278.
- Touya, G., Bucher, B., Falquet, G., Jaara, K. and Steiniger, S. (2014) Modelling Geographic Relationships in Automated Environments. In Burghardt, D., Duchene, C. and Mackaness, W. A. (eds.) *Abstracting Geographic Information in a Data Rich World*. Heidelberg: Springer, pp. 53-82.

- Touya, G., Balley, S., Duchene, C., Jaara, K., Regnauld, N. and Gould, N. (2012) Towards an Ontology of Spatial Relations and Relational Constraints. *In 15th Workshop of the ICA commission on Generalisation and Multiple Representation*. Istanbul, Turkey, 13th-15th September 2012.
- Trafford Council. (2012) *DataGM*. [Online] [Accessed on 12th June 2014] <http://datagm.org.uk/>
- Tsarkov, D., Horrocks, I., Furbach, U. and Shankar, N. (2006) Fact++ description logic reasoner: System description. *In Furbach, U. and Shankar, N. (eds.) Automated Reasoning*. Vol. 4130. Springer Berlin / Heidelberg, pp. 292-297.
- Tudorache, T., Nyulas, C., Noy, N. F. and Musen, M. A. (2013) WebProtégé: A collaborative ontology editor and knowledge acquisition tool for the Web. *Semantic Web*, 4(1), pp. 89-99.
- University of Edinburgh. (2014) *Digimap*. [Online] [Accessed on 12th June 2014] <http://digimap.edina.ac.uk/>
- University of Oxford. (2011) *Hermit*. [Online] [Accessed on 12th June 2014] <http://hermit-reasoner.com/index.html>
- Uschold, M. and King, M. (1995) Towards a methodology for building ontologies. *In Proceedings of IJCAI95's Workshop on Basic Ontological Issues in Knowledge Sharing*.
- van Oosterom, P. and Meijers, M. (2011) Towards a true vario-scale structure supporting smooth-zoom. *In 14th Workshop of the ICA commission on Generalisation and Multiple Representation*. Paris, France, 30th June - 1st July 2011.
- van Rees, R. (2003) Clarity in the usage of the terms ontology, taxonomy and classification. *CIB REPORT*, 284 p. 432.
- Varanka, D. and Caro, H. (2013) Spatial Relation Predicates in Topographic Feature Semantics. *In Raubal, M., Mark, D. M. and Frank, A. U. (eds.) Cognitive and Linguistic Aspects of Geographic Space*. Springer Berlin Heidelberg, pp. 175-193.
- Visvalingam, M. and Whyatt, J. D. (1993) Line generalisation by repeated elimination of points. *The Cartographic Journal*, 30(1) pp. 46-51.
- Wang, Z. and Müller, J. C. (1998) Line Generalization Based on Analysis of Shape Characteristics. *Cartography and Geographic Information Science*, 25(1) pp. 3-15.
- Weibel, R. (1997) Generalization of spatial data: Principles and selected algorithms. *In Kreveld, M., Nievergelt, J., Roos, T. and Widmayer, P. (eds.) Algorithmic Foundations of Geographic Information Systems*. Vol. 1340. Springer Berlin/Heidelberg, pp. 99-152.
- Weibel, R., Keller, S. and Reichenbacher, T. (1995) Overcoming the knowledge acquisition bottleneck in map generalization: The role of interactive systems and computational intelligence. *In Frank, A. U. and Kuhn, W. (eds.) Spatial Information Theory A Theoretical Basis for GIS*. Vol. 988. Springer Berlin Heidelberg, pp. 139-156.
- Weiss, R. and Weibel, R. (2013) Road Network Selection for Small-Scale Maps Using an Improved Centrality Approach. *In 16th Workshop of the ICA commission on Generalisation and Multiple Representation*. Dresden, 23rd–24th August 2013
- Wielinga, B. J. (2013) Reflections on 25+ years of knowledge acquisition. *International Journal of Human-Computer Studies*, 71(2) pp. 211-215.
- Wilson, D., Bertolotto, M. and Weakliam, J. (2010) Personalizing map content to improve task completion efficiency. *International Journal of Geographical Information Science*, 24(5), 2012/07/04, pp. 741-760.

Wolf, E. (2009) Ontology-Driven Generalization of Cartographic Representations by Aggregation and Dimensional Collapse. In Bernstein, A., Karger, D., Heath, T., Feigenbaum, L., Maynard, D., Motta, E. and Thirunarayan, K. (eds.) *The Semantic Web - ISWC 2009*. Vol. 5823. Springer Berlin / Heidelberg, pp. 990-997.

Xavier, D., Morán, F., Fuentes-Fernández, R. and Pajares, G. (2013) Modelling knowledge strategy for solving the DNA sequence annotation problem through CommonKADS methodology. *Expert Systems with Applications*, 40(10) pp. 3943-3952.

Xu, J., Nyerges, T. L. and Nie, G. (2014) Modeling and representation for earthquake emergency response knowledge: perspective for working with geo-ontology. *International Journal of Geographical Information Science*, 28(1) pp. 185-205.

Yan, H. and Weibel, R. (2008) An algorithm for point cluster generalization based on the Voronoi diagram. *Computers and Geosciences*, 34(8) pp. 939-954.

Yan, H., Weibel, R. and Yang, B. (2008) A Multi-parameter Approach to Automated Building Grouping and Generalization. *Geoinformatica*, 12(1) pp. 73-89.

Zhou, Q. and Li, Z. (2012) A comparative study of various strategies to concatenate road segments into strokes for map generalization. *International Journal of Geographical Information Science*, 26(4) pp. 691-715.

Zhou, S. and Jones, C. (2003) A Multi-representation Spatial Data Model. In Hadzilacos, T., Manolopoulos, Y., Roddick, J. and Theodoridis, Y. (eds.) *Advances in Spatial and Temporal Databases*. Vol. 2750. Berlin / Heidelberg: Springer, pp. 394-411.

Appendix A

Paper presented to the Geoprocessing 2012 conference, Valencia, Spain, 30th January – 4th February 2012.

Generation and Validation of Workflows for On-demand Mapping

Nick Gould and Omais Chaudhry

Manchester Metropolitan University

Manchester, UK emails: nicholas.m.gould@stu.mmu.ac.uk, o.chaudhry@mmu.ac.uk

Abstract— The paper presents a method to automatically select and sequence the tasks required to build maps according to user requirements. Workflows generated are analysed using Petri nets to assess their validity before execution. Although further work is required to select the optimal method for generating the workflow and to execute the workflow, the proposed method can be used on any workflow to assess its validity.

Keywords— automated map generalisation; workflows; Internet mapping; Petri nets.

I. INTRODUCTION

The development of Google Maps and similar products has led to a vast number of ‘mashups’ where users can overlay their own data on Google Maps backgrounds and make the resultant map available to others. The problem with this approach is that the user is limited to the background maps supplied by Google; there is no, or very little, flexibility to vary the content depending on the context and there is no data integration [1]. This is highlighted in Fig. 1 where the street names are obscured by overlaid cycle routes. Further problems may occur when the scale changes. For instance, a minor road that may be part of a cycle route may disappear at smaller scales since the two datasets are independent.

What is required is a system to allow data from a variety of sources to be mapped at a variety of scales. Since, the possible combination of datasets and scales is too numerous to be pre-defined, on-demand generalisation (deriving smaller scale maps from larger scale maps) is necessary.

Cartographic generalisation is a complex process [2] and much effort has gone in to developing automation techniques that reduce or eliminate human involvement [3].

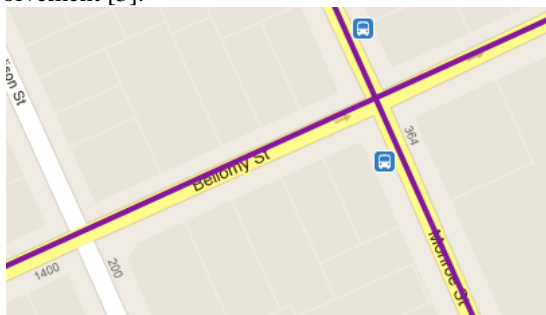


Figure 1 Google Maps with cycle routes overlaid

The focus has, until recently, been on allowing National Mapping Agencies (NMAs) to automate the production of maps at different scales from a single master source [4][5]. Automatic generalisation is applied to a pre-defined set of map features at pre-defined scales to produce a pre-defined set of products. However, the advance of neo-geography and Volunteered Geographic Information [6] means that on-demand generalisation is required allowing users to integrate their data with that of NMAs and other mapping resources. There have been attempts to generate online on-demand maps to user requirements, but such systems have been developed by applying a fixed sequence of generalisation operations to known datasets [7][8].

An on-demand mapping system will require a number of components including a means of taking high level user requirements (e.g., “I want a city-wide map of road accidents”) and producing a machine-readable specification of the map [9]. The system will also need a knowledge base to store cartographic rules of the type: “if the scale is greater than 1:30,000 omit minor roads”. A set of map generalisation services are then required to satisfy such rules or constraints. Traditionally the selection of map generalisation operators and their sequencing is done by cartographic experts, but for on-demand mapping, aimed at the non-expert user, a system is required that can automatically generate, validate, execute and monitor these operations; in other words a workflow needs to be generated and executed [9]. The focus of this research is on developing a workflow engine that, given the specification, using the rules, will automatically select, sequence, and execute the map generalisation services required to generate the map or spatial output.

This paper describes the initial attempts to automatically generate a workflow for building a map based on user requirements and suggests how to validate that workflow.

To illustrate the process, a use case involving the mapping of road accidents will be employed. Fig. 2 represents a detailed map of accidents at a road junction.

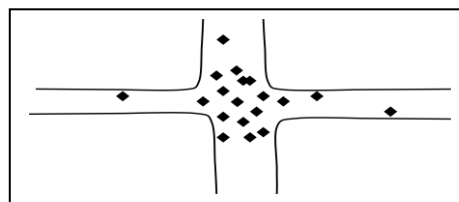


Figure 2 Accidents at a road junction

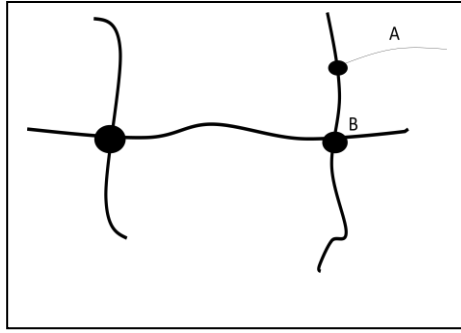


Figure 3 Generalised data at a small scale

To represent all of the data at a smaller scale the road network is generalised by *eliminating* any minor roads and *collapsing* (reducing to single lines) the major roads. To avoid information overload the accidents are *clustered* (Fig. 3). Elimination, collapse and clustering are common generalisation operations. The junction in Fig. 2 is represented by 'B'.

The generalised map serves to highlight accident hot spots. However, by removing the minor road 'A', context is lost, since the cluster will appear to be on a straight section of road, so a step is required to reinstate those minor roads that intersect a cluster. What we have is a set of tasks that have to be carried out, some of which are in a particular order, i.e., we cannot reinstate the minor roads until we have created the clusters. Since there a number of tasks to execute, some of which have to be completed before others can start, a workflow is required to express these relationships. In addition that workflow has to be valid, i.e., all of the tasks must, at least, be called.

The method used was based on the premise that workflow definitions can be analysed using Petri nets for flaws that would stop the workflow from completing execution [10][11].

Firstly, a technique was implemented to generate the list of tasks and their dependencies based on applying user requirements to a set of applicable rules (described in Section II). From this a workflow definition could be created, represented by a directed graph (Section III). A method was developed to produce a Petri net from a given directed graph (Section IV). The Petri net could then be analysed for flaws in the workflow definition (Section V).

II. GENERATING A LIST OF TASKS AND DEPENDENCIES

There are a number of different techniques for automatically generating workflows including Case Based Reasoning [12] and product structures, where the map is treated as a product that has to be constructed from component parts [13].

The technique used in this research was one that employs user preferences to define the selection and execution of a set of rules [14]. This technique was selected because of its focus on the user's needs, which is essential for on-demand mapping.

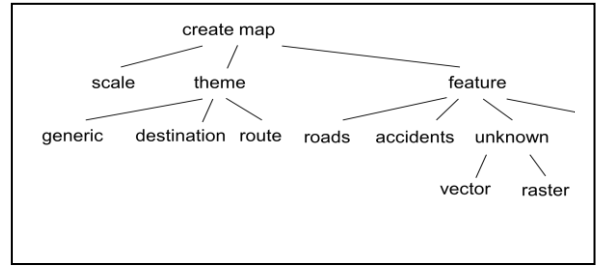


Figure 4 Knowledge hierarchy

The user preferences are gathered by navigating a knowledge/rule hierarchy (Fig. 4). If a particular branch is not selected by the user than that branch is closed off. For example, if the user does not select an 'unknown' feature type they are not prompted for 'vector' or 'raster'. In the prototype the user is simply prompted for his or her preferences using text boxes and drop down boxes in a web page. Each leaf node in the hierarchy has one or more associated rules; these are added to a set of applicable rules as the user requirements are gathered.

If the user selects the 'roads' feature type then the rules associated with that feature type (R1, R2, R3) are added to the set of applicable rules. Rules consist of a condition and an action (e.g., *insert* a task to the workflow or *order* two tasks) and are stored in an XML file (Fig. 5). Using XML allows for the use of schemas to enforce correct structure.

The gathered user requirements are held in memory as ordered pairs, for example:

```
< scale,50000 >
< theme,generic >
< featureType,roads >
< featureType,accidents >
```

```
...
<featureType name="roads">
  <rules>
    <rule id="R1">
      <condition>scale >= 5000 AND
featureType = roads</condition>
      <action>insert(t1)</action>
    </rule>
    <rule id="R2">
      <condition>scale >= 5000 AND
featureType = roads</condition>
      <action>insert(t2)</action>
    </rule>
    <rule id="R3">
```

Figure 5 Knowledge/rule hierarchy (partial) as XML

Once these have been collected, the applicable rules are then evaluated, checking rule conditions against user preferences to generate a set of tasks and a set of task dependencies. For example, if the user has selected the feature type “roads” and a map scale of greater than 1:5000 then the conditions for rules R1, R2 and R3 (Fig. 5) will be met and their actions triggered, e.g. task t1 is inserted into the workflow. The action ‘order(t2,t1)’ means task t1 is a dependency of task t2 and should only be run after t2 has been executed.

Using the above use case, the selected tasks may be:

t1: collapse roads
t2: delete minor roads
t9: add copyright notice for roads dataset
t3: cluster accidents
t8: reinstate minor roads on clusters

and the dependencies:

t2 < t1
t1 < t8
t3 < t8

In this case, there are five tasks to perform and there are three dependencies (or precedence constraints). For example, we want to delete any minor roads (task t2) before we collapse the roads (task t1) as it is inefficient to collapse a subset of the road network that we are later going to delete. Task t9 is not involved in any dependency and is classified as an independent task.

The above output can be expressed as a directed graph where tasks are represented as nodes and dependencies as edges (Fig. 6).

However, the graph does not constitute a workflow. The next section describes why and then what is needed to construct a workflow definition.

III. CREATING A WORKFLOW DEFINITION

The directed graph (Fig. 6) generated from the example set of task dependencies does not make up a workflow definition. This is because there is no place for any independent tasks (in our case task t9). In addition, the following rules must be satisfied for a workflow definition according to [15]:

- The workflow graph should have a single source node and a single sink node
- Every other node should have at least one parent and at least one child

This ensures that the workflow has a defined start and end and that there are no unnecessary tasks.

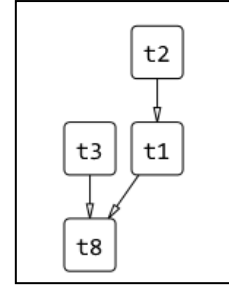


Figure 6 Directed graph based on dependencies

A workflow definition directed graph can be created by the following procedure:

1. Add start (A) and end (B) nodes
2. Create an edge for every dependency
3. For every node that has no children add an edge to the end node
4. For every node without a parent add an edge from the start node
5. For each independent tasks link the task directly to the start and end node.

The revised graph can be seen in Fig. 7.

The method so far has produced a workflow definition for the given case study but is it valid? For instance, it is relatively easy to ensure that there are no directly contradictory dependencies between the selected tasks so that both t1 < t2 and t2 < t1 did not appear in the same workflow. However, indirectly contradictory dependencies such as that seen in Fig. 8 would be harder to identify. In this example the dependency “t3 precedes t14” has introduced *deadlock* [15] into the workflow. Task t3 will not start until t8 starts; but t8 will not start until t14 starts, which will not start until t3 starts. So, tasks t14, t8, t5, t3 and subsequent tasks will never be executed.

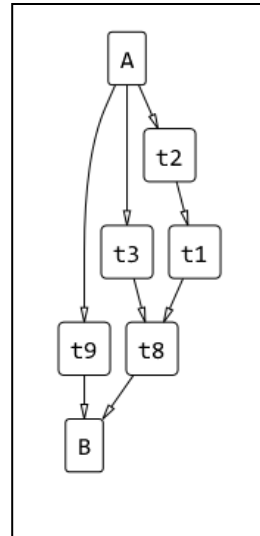


Figure 7 Workflow definition graph

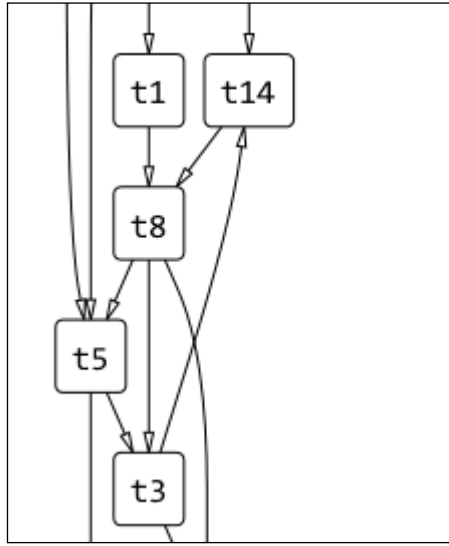


Figure 8 Deadlock in a workflow

A method of testing the soundness of a workflow before attempting execution is needed. The simplest way of checking for deadlock is by performing a topological sort on the graph. However, the application of Petri nets will allow for a more expressive form of graph and the ability to describe more complex workflow patterns [16] than that described above. In addition to describing workflows, the *mathematical foundations* of Petri nets [17] allow for the analysis of workflows and are applicable to more complex analysis than the deadlock problem [10][11]. Extensions to Petri nets, such as coloured Petri nets, which allow for the investigation of delays and throughput, have been defined formally [11]. Petri nets offer a number of advantages over PERT charts such as the ability to model nondeterministic behaviour and loops in the workflow [31]. The adoption of Petri nets at an early stage will allow the design to be scaled to more complex workflows. But, first, we need to generate the Petri net from the directed graph.

IV. GENERATING A PETRI NET

A Petri net is a particular class of directed graph, defined as a bipartite directed graph consisting of two types of nodes called transitions and places [17]. Nodes are linked by arcs such that arcs cannot link a place to a place or a transition to a transition. Transitions, denoted by rectangles, represent events or, in our case, workflow tasks. Places denoted by circles, represent states (Fig. 9).

Staines [18] describes the process for generating a directed graph from a Petri net, which can be reversed to generate a Petri net. The procedure used is as follows:

1. Nodes (tasks) are converted to transitions
2. Each edge generates arc-place-arc
3. Extra places are added preceding the start node (A) and following the end node (B).

The Petri net generated from the workflow shown in Fig. 7 can be seen in Fig. 9.

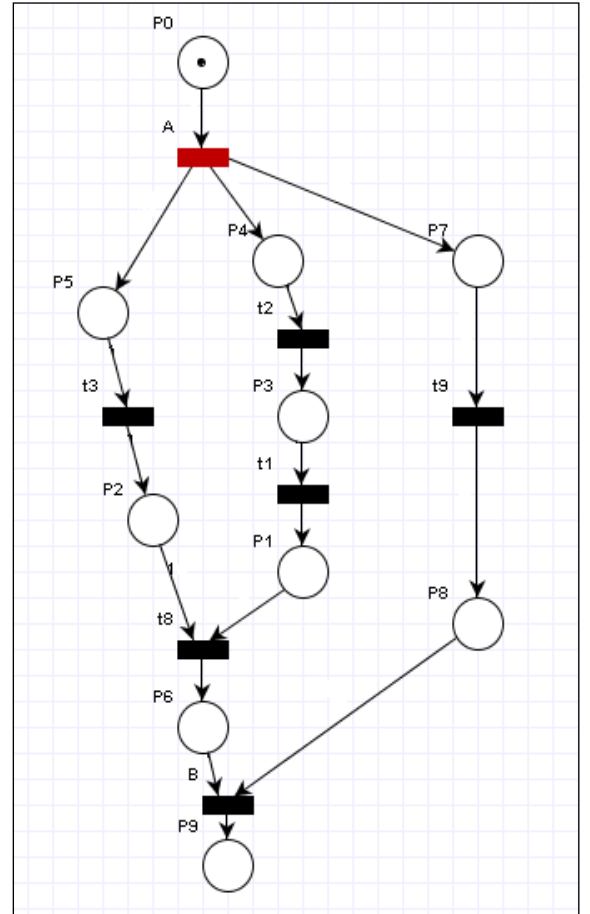


Figure 9 Petri net for valid workflow

The starting place, P0, contains a single *token*. Tokens can be used to model the workflow. A transition may be fired only if there are one or more tokens in all of its input places [17]. In this example, transition A can be fired. When a transition fires it takes a token from each of its input places and places a token in each of its output places. So after the firing of transition A, there will be a token in each of the places P5, P4, P7 (but no longer P0) thus enabling transitions t3, t2 and t9. Note that t8 will not be fired until both t3 and t1 have, which models the original dependencies.

Code was written to generate an XML file in a format that can be read by the PIPE software [19]. PIPE can then be used to visualise and animate the Petri net firings to ascertain whether the workflow is executable.

Deadlock can be identified visually or by using a Petri net analysis tool such as PIPE. It needs, however, to be identified as part of the on-demand mapping system. The following section describes how this was done.

V. VALIDATING THE WORKFLOW

Our system generates a *workflow net* [15], a particular type of Petri net such that:

- The net has a single source and a single sink node

- Every task lies on a directed path between the source and the sink nodes.

However, as has been shown, a workflow net containing anomalies such as deadlocks can still be generated. A *sound* process is defined as one that contains no unnecessary tasks and where every case submitted to the process must be completed in full and no reference to it remaining in the process, i.e., for every token that is in the start place there is one token in the end place and no others in the net [15].

There is a number of, sometimes complex, techniques for checking the soundness of a workflow net. Fortunately the Petri nets derived from our workflow generation are a particular sub-class of Petri nets known as conflict free or T-systems where every place has no more than one input and one output transition [20]. In effect conflicts are ruled out; there are no logical ORs in the system. This makes them easier to analyse [21].

In the prototype the Petri net is checked for deadlock by looping through the set of places and firing any transitions that are enabled until no more transitions can be fired. If all the transitions are fired then the workflow is valid, if there are transitions that cannot be fired then these are listed.

In addition to the case study, the method was tested on a number of randomly generated task lists and dependencies. A demonstration version of the prototype can be seen at www.ondemandmapping.org.uk (Fig. 10).

VI. CONCLUSION AND FUTURE WORK

The increasing availability of once inaccessible datasets and the explosion of crowd-sourced data, alongside the growth of web-based mapping, have led to the need for on-demand mapping. The requirement to integrate data from a number of disparate sources means that there is a need to automate the creation of the workflow required to generate such maps.

Figure 10 Prompting for user requirements

This paper has presented two aspects of automatic workflows; firstly the generation of the workflow from simple user specifications and secondly the generation

of Petri nets from the workflow definitions to allow for their validation. In particular the work done so far has highlighted the potential problem of contradictory rules that can generate deadlocks in workflow definitions.

It was assumed that the generation of the workflow is a static scheduling problem, i.e., the workflow is deterministic, known in advance of execution [22]. This is likely to be a simplification of the on-demand mapping problem; it will be necessary to consider how the workflow may change during execution when, e.g., a particular generalisation service is not available at execution time. For this reason adaptive and autonomic workflow techniques [23][24][25] may need to be investigated. However, it could be argued that any replacement service or set of services would not affect the workflow if the replacement(s) could be represented as a sub-net with a single point of entry and a single point of exit to replace the failed service.

Further work is also required on the means for expressing the cartographic rules. For example, in the case study (Fig. 5) three rules had the same conditions but different actions. Could the rules be expressed more concisely? Also required is further investigation into how the rule base is to be populated and the knowledge hierarchy defined.

The execution of workflows will consist of calling a number of web services that provide generalisation operators. Web services are usually orchestrated using Business Process Execution Language (BPEL) [26]. Once sound workflow nets can be generated and validated using Petri nets it will be useful to investigate the process of generating BPEL from Petri nets [27][28].

Previous research into the orchestration of generalisation services in particular [29][30] will also need to be considered with a view to investigating how to integrate such services into the system.

A major problem with the work done so far is the lack of a data model. The method lacks the concept of tasks doing work on spatial datasets. Datasets have to be managed as they progress through the workflow and conflicts have to be handled when two different tasks attempt to work on the same dataset at the same time. One possibility may be to regard the presence of a dataset as a pre-condition to the firing of a transition. The transition would not fire until the dataset was available. The output from the transition would then be the processed dataset, e.g., a set of clustered accidents.

Whatever the eventual process is employed for generating the workflow, it is believed that the method described in this paper can be used to validate the workflow definition before an execution is attempted.

ACKNOWLEDGEMENTS

This project is funded by the Dalton Research Institute at Manchester Metropolitan University and the Ordnance Survey of Great Britain. Thanks to Martin Stanton of MMU for guidance on the use Petri nets.

REFERENCES

- [1] J. Gaffuri, "Improving web mapping with generalization," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 46, no. 2, January 2011, pp. 83-91.
- [2] E.M. João, *Causes and consequences of map generalisation*. London: Taylor and Francis, 1998.
- [3] Z. Li, "Digital map generalization at the age of enlightenment: A review of the first forty years," *Cartographic Journal*, vol. 44, no. 1, February 2007, pp. 80-93.
- [4] J. Stoter, et al., *State-of-the-art of automated generalisation in commercial software*. 2010. Available from: http://www.euroedr.net/projects/generalisation/euroedr_gen_final_report_mar2010.pdf 01.09.2011
- [5] A. Ruas, and C. Duchêne, "A prototype generalisation system based on the multi-agent system paradigm," in *Generalisation of Geographic Information*, A.M. William, R. Anne, and L.T. Sarjakoski, Eds., Amsterdam: Elsevier Science, 2007.
- [6] M. Goodchild, "Citizens as sensors: the world of volunteered geography," *GeoJournal*, vol. 69, no. 4, 2007, pp. 211-221.
- [7] F. Grabler, M. Agrawala, R. W. Sumner and M. Pauly, "Automatic generation of tourist maps," *Proc. ACM SIGGRAPH 2008 papers*, Los Angeles, California: ACM, 2008.
- [8] K. Johannes, A. Maneesh, D. Barger, S. David and M. Cohen, "Automatic generation of destination maps," *ACM Trans. Graph.*, vol. 29, no. 6, December 2010, pp. 1-12.
- [9] S. Balley and N. Regnaud, "Collaboration for better on-demand mapping," in *ICA Workshop on Generalisation*, Paris, France, 2011.
- [10] N. R. Adam, V. Atluri and W. K. Huang, "Modeling and analysis of workflows using Petri Nets," *Journal of Intelligent Information Systems*, vol. 10, no. 2, March/April 1998, pp. 131-158.
- [11] W. M. P. van der Aalst, "The application of Petri nets to workflow management," *Journal of Circuits, Systems and Computers*, vol. 8, no. 1, 1998, pp. 21-66.
- [12] A. Aamodt and E. Plaza, "Case-based reasoning: foundational issues, methodological variations, and system approaches," *AI Communications*, vol. 7, no. 1, 1994, pp. 39-59.
- [13] W. M. P. van der Aalst, "On the automatic generation of workflow processes based on product structures," *Computers in Industry*, vol. 39, no. 2, 1999, pp. 97-111.
- [14] S. Chun, V. Atluri and N. Adam, "Domain knowledge-based automatic workflow generation," *Proc. Database and Expert Systems Applications*, 13th International Conference, Aix-en-Provence, France, Berlin: Springer, 2002, pp. 778-838.
- [15] W. M. P. van der Aalst and K. M. van Hee, *Workflow management models, methods, and systems*, Cambridge, Mass.: MIT, 2004.
- [16] N. Russell, A. ter Hofstede, W. van der Aalst, and N. Mulyar, "Workflow Control-Flow Patterns: A Revised View," *Technical Report BPM-06-22*, 2006; Available from: <http://www.workflowpatterns.com> 01.09.2011
- [17] T. Murata, "Petri nets: properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, 1989, pp. 541-580.
- [18] A. S. Staines, "Rewriting Petri Nets as Directed Graphs," *International Journal of Computers*, vol. 5, no.2, 2011, pp. 289-297
- [19] N. Akharware, *PIPE - Platform Independent Petri net Editor* 2. 2005; Available from: <http://pipe2.sourceforge.net/> 01.09.2011
- [20] J. Desel and J. Esparza, *Free Choice Petri Nets*, Cambridge: Cambridge University Press, 1995.
- [21] P. Alimonti, E. Feuerstein, U. Nanni and I. Simon, "Linear time algorithms for liveness and boundedness in conflict-free Petri nets," in *LATIN '92 Lecture Notes in Computer Science*, Berlin: Springer, 1992, pp. 1-14.
- [22] J. W. Herrmann, C.-Y. Lee, and J. L. Snowdon, "A classification of static scheduling problems," in *Complexity Issues in Numerical Optimization*, P. M. Pardalos, Ed, World Scientific Publishing Co.: Singapore, 1993, pp. 203-253.
- [23] R. Muller, U. Greiner, and E. Rahm, "AgentWork: a workflow system supporting rule-based workflow adaptation," *Data & Knowledge Engineering*, vol. 51, no. 2, 2004, pp. 223-256.
- [24] M. Polese, G. Tretola and E. Zimeo. "Self-adaptive management of Web processes," *Proc. Web Systems Evolution (WSE)*, 12th IEEE International Symposium, 2010.
- [25] G. Tretola, and E. Zimeo, "Autonomic internet-scale workflows," *Proc. of the 3rd International Workshop on Monitoring, Adaptation and Beyond*, Ayia Napa, Cyprus, New York: ACM, 2010.
- [26] OASIS. *Web Services Business Process Execution Language v2.0*. 2007; Available from: <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf> 01.09.2011
- [27] P. Sun, C. Jiang and M. Zhou, "Interactive Web service composition based on Petri net," *Transactions of the Institute of Measurement and Control*, vol. 33, no. 1, 2011, pp. 116-132.
- [28] W. M. P. van der Aalst and K.B. Lassen, "Translating unstructured workflow processes to readable BPEL: Theory and implementation," *Journal of Information Software Technology*, vol. 50, no. 3, 2008, pp. 131-159.
- [29] T. Foerster, L. Lehto, T. Sarjakoski, L. T. Sarjakoski, and J. Stoter, "Map generalization and schema transformation of geospatial data combined in a Web Service context," *Computers, Environment and Urban Systems*, vol. 34, no. 1, 2010, pp. 79-88.
- [30] G. Touya, C. Duchêne, and A. Ruas, "Collaborative generalisation: formalisation of generalisation knowledge to orchestrate different cartographic generalisation processes," *Proc. of the 6th international conference on Geographic Information Science*, Zurich, Berlin: Springer-Verlag, 2010.
- [31] D. Dubois, K. Steck, "Using Petri nets to represent production processes," *Proc. of the 22nd IEEE Conference on Decision and Control*, 1983.

Appendix B

Paper presented to the 1st AGILE PhD School, Wernigerode, Germany, 13th - 14th March 2012.

Semantic description of generalisation web services for on-demand mapping

Nicholas Gould, M.Sc., Phone: ++44.161431-7681, E-Mail: nicholas.m.gould@stu.mmu.ac.uk, Manchester Metropolitan University, Oxford Road, M15 6BH, Manchester, UK

Abstract

This paper discusses the need for the semantic description of web services that implement map generalisation algorithms to allow the services to be selected automatically. Particular focus is placed on need to define a common set of parameters that can be used by any number of algorithms.

Keywords: automatic generalisation; semantic web services; on-demand mapping

1. Introduction

This project focuses on automated mapping techniques. The development of Google Maps has led to a vast number of “mashups” where users can overlay their own data on a Google Maps background and make the result available to others. The problem with this approach is that the creator is limited to using the background map as supplied and there is no opportunity to vary the content depending upon the context. The lack of integration of user-supplied data leads to cartographic problems such as road names being obscured by overlaid cycle routes, for example (Figure 1).

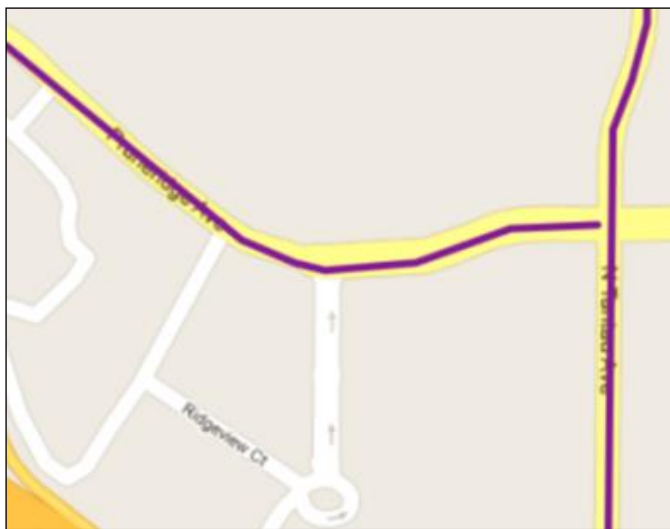


Figure 1 Cycle routes on Google maps

What is required is a system to allow data from a variety of sources to be mapped at a variety of scales. Since the possible combination of datasets and scales is too numerous to be pre-defined, on-demand generalisation (deriving smaller scale maps from larger scale maps) integrating user-supplied data is needed. There have been attempts to provide online on-demand maps (Kopf, et al., 2010) but such systems have been developed by applying a fixed sequence of generalisation operations to a known set of data.

The aim of the project is to aim is to design a system for the automatic generation of workflows for on-demand mapping. This will involve the automatic selection, sequencing and execution of generalisation web services based on the OGC's Web Processing Service (WPS) (Open Geospatial Consortium, 2011).

The steps involved in creating and executing the workflow are *abstract composition*, *concrete composition*, and *execution*.

The abstract composition phase will involve taking the user's requirements and the descriptions of the data sources and generating an abstract workflow that consists of generalisation operators (simplification, smoothing, amalgamation etc.) The concrete composition phase will involve finding the appropriate web services for those operators and the execution phase will involve calling and managing those services. This paper focusses on the concrete composition phase and in particular two issues; 1) how to annotate the web services in such a way that they can be automatically selected, and 2) once a service has been selected, to supply values to its parameters automatically.

1.1 Definitions

A single WPS service can perform a number of *processes*. Each process will implement one or more generalisation algorithms (atomic or composite processes). An algorithm may implement one generalisation operator (e.g., building *amalgamation*) or a combination of operators (e.g., road *simplification* and *smoothing*).

1.2 The need for the semantic description of services

The selection and sequencing of the generalisation operators, to build the abstract workflow, will be derived from the system's cartographic knowledge, for example, generalisation of a road network might require *selection*, *simplification* and *smoothing* in that order. The issue is how do we select the appropriate web services to perform these generalisation operators (concrete workflow)? We need semantic descriptions of the web services to allow for the automatic selection and parameterisation of algorithms. However, it is a well-documented problem with the OGC protocols that they provide for *syntactic* interoperability but not *semantic* interoperability (Janowicz et al., 2010; Lemmens et al., 2007). A free-text description field is not sufficient to allow for the automatic selection of a process.

The second issue is, once selected, how do we execute the services i.e. supply the required parameter values? WPS services are syntactically interoperable in that for any parameter a data type is defined (double, integer etc.). This is not sufficient; we need to know the meaning of the parameters. For example, the point clustering ISODATA algorithm (Li, 2007) has two parameters both of which represent a distance – how will the system be able to distinguish between the two?

Another problem is that some algorithms will have geographically meaningless parameters. The 'simulated annealing' approach to automatic feature displacement includes among its parameters an 'initial temperature' and a 'temperature gradient' (Ware et al, 2003). How can values for these parameters be derived from user requirements expressed in geographic terms such as target scale? Other algorithms such as the Douglas-Peucker line simplification have a single parameter, 'tolerance', which has a geographic meaning i.e., the maximum distance from the original line and the simplified line (Douglas and Peucker, 1973). However, the Visvalingam-Whyatt algorithm, which also performs line simplification, has a different concept of tolerance, a minimum area (Visvalingam and Whyatt, 1993).

Some algorithms define as many as six parameters (Revell, 2004; Ware et al 2003) others have none (Yan and Weibel, 2008). Every time a new algorithm was exposed by a web service the system's knowledge would have to be updated. Rather than the system having knowledge of all of the parameters for every algorithm it may use, it is more sensible for the system to define a *common set of parameters* that all algorithms must use. It would be the responsibility of the service to translate from the common parameters to the specific parameters of the algorithm.

In summary, processes need to be described in such a way that the system selects the most suitable WPS process for each task and then provides appropriate values for their parameter. But what information is required to describe a service sufficiently? Section 3 describes a classification of generalisation algorithms and their parameters. Section 4 includes a survey of generalisation algorithms based on this classification. Section 5 discusses the conclusions and future work.

2. Related work

How are the generalisation web services to be semantically enhanced? One approach is *semantic annotation* (Maue et al., 2009; STI Innsbruck, 2011) where OGC standards are enhanced with semantic descriptions. A comprehensive approach is the *semantic enablement layer* (Janowicz et al, 2010) that includes a *web ontology service* that manages a processing ontology and a features ontology, and a *web-reasoning service* that encapsulates a semantic reasoner that aims to combine processing and features to meet the user's needs.

2.1 Defining a common parameter set

Touya et al (2010) describe *translator functions* that take the set of constraints and set of rules held in the system knowledge and the user requirements and translates them into algorithm parameters. Each algorithm will have its own translator function. This approach embeds the translator component in the system rather than ask each generalisation service to manage its own translation.

When considering line simplification Foerster et al. (2007) defined a *simplification ratio* that could be used by both the Douglas-Peucker and the Visvalingam-Whyatt algorithms. The value for the ratio is dependent on the user's choice of target scale and is based on the ratio of the number of vertices before and after simplification. The ratio is a variation on the Radical law of selection (Töpfer and Pilliwizer, 1966) which is used to determine the number of objects to be retained on a map at reduced scale based on the number of source objects and the ratio of the source and target scales.

The Radical law is also used in a point generalisation algorithm (Yan and Weibel, 2008) as a limit on the number of iterations the algorithm performs. The same algorithm considers the importance value of the source points when deciding to retain a point. By employing the user requirements (target scale) and the source data (number of points and importance values) this algorithm requires no parameters. Could these concepts be used to provide parameter values for those generalisation algorithms that do have parameters?

3. Classification of Algorithms

Following discussions with the project's partner, the Ordnance Survey of Great Britain, a set of attributes that were needed to describe generalisation algorithms, and hence any web service that implemented them, was defined and extended (Table 1).

Attribute	Description
Description	Free text description of what the algorithm does (this will not be part of the algorithm's semantic description but will aid its construction).
Operators	The generalisation operator(s) that the algorithm implements. This needs to be classified.
Feature types	The feature type that the process applies to; e.g. buildings. Some algorithms are either specific to, or work best on, particular feature types. We will need a classification of feature types.
Geometry of the input data	We need to know what type of data the algorithm uses.
Scale related information	Maxima and minima for <i>target scale</i> and <i>scale ratio</i> .
Nature of source data	Some algorithms work better on rural buildings than they do on urban buildings, for example. This will have to be classified.
Data quality	In particular data input quality – e.g. for algorithms operating on networks issues such as a badly formed network or duplicate vertices can cause problems.
Performance	Some algorithms may be restricted in the size of input dataset they can operate on; e.g., a maximum number of features.
Time factor	Some algorithms may produce high 'quality' results but take a relatively long time and vice versa (the selection of such services will be dependent on user requirements). This needs to be quantified.

Table 1 Algorithm descriptions

Model generalisation	Cartographic generalisation
Class selection	Enhancement
Reclassification	Displacement
Collapse	Elimination
Combine	Typification
Simplification	
Amalgamation	

Table 2 Operator classification (Foerster et al, 2007)

Since the abstract workflow will be based on generalisation operators we will need to classify algorithms by the operator(s) they implement and for that we will need a

consistent way of classifying operators; do *filtering* and *elimination* mean the same thing, for example? The most recent attempt to classify operators was by Foerster et al (2007) (Table 2) which makes a distinction between model and cartographic generalisation. However, this classification is at too high a level of abstraction; for example, *enhancement* includes *exaggeration*, *smoothing* and *enlargement* so it is not sufficient to state that a road network requires *enhancement*, we need to be more specific.

A complement to the generalisation operator method of classifying algorithms would be to adopt the descriptions used by Li (2007) in his survey, where the nature of the source data is incorporated in the description, hence we have categories such as 'smoothing individual line features' and 'transformation of individual area features' (e.g. building simplification). Li introduces sub-categories that refer to the processing method so 'transformation of individual area features' can have the sub-categories 'Boundary-based shape simplification of an area feature' and 'Region-based shape simplification of an area feature'. Also included are subcategories that relate to the generalisation operator they implement such as 'Collapse of area features'. Although the categorisation used by Li has not been formalised it offers the advantage of describing what action is being taken on what type of data and is thus relatively expressive.

3.1 Description of parameters

A method of describing parameters is required such that the workflow system can automatically provide values for the parameters based on the user requirements and the nature of the data that is to be generalised. A number of attributes for describing parameters was defined (Table 3) based on the requirement to describe a parameter sufficiently that values can be supplied automatically. The next step was to apply the classifications to a set of generalisation algorithms.

4. Survey of generalisation algorithms

The algorithms and their parameters were described using the categories listed above (Table 1, Table 3). It was decided to focus on point clustering algorithms, line simplification and smoothing, and building simplification and amalgamation. This includes point, line and polygon data. Table 4 provides an example description.

To give an indication as to the variety of parameters, the name of each parameter was extracted from the survey data and represented as a word cloud (Figure 2).

The survey has exposed the large variety of parameters in just a subset of the generalisation operators. The word cloud serves to highlight certain parameters such as *minimum distance* that appear frequently. However, the meaning of what minimum distance represents varies from algorithm to algorithm.

Attribute	Description
Name	Name of the parameter e.g. <i>maximum distance</i>
Description	Free text description of the parameter (this will not be part of the algorithm's semantic description but will aid it).
Class	What the parameter represents. E.g. a <i>tolerance</i>
Effect on output	Describes the particular impact of the parameter.
Weight	How important is this parameter in relation to other parameters of the algorithm? This is a quantification of the previous parameter
Necessity	Can the parameter be omitted or can a default be provided? This needs to be codified as a Boolean attribute.
Data type	Double, integer etc.
Units	Linear units, areal units etc.
Range	Range of possible values for the parameter. This can be used as a check against non-sensible values being supplied.

Table 3 Description of parameters

Algorithm	ISODATA							
References	Li, 2007, p78							
Li Category	Aggregating point features (clustering)							
Operator(s)	Combine							
Description	Repeatedly split a cluster until the maximum standard deviation of distance from centre in each direction (x,y) is reached.							
Application	Punctual events							
Source data	Set of points							
Parameters								
Name	Description	Class	Effect on output	Necessity	Weight	Data type	Units	Range
$\sigma_{x,max}$	Max. standard deviation from centre of any cluster in x direction	Maximum Distance	Fixes the maximum width of any cluster	Required	0.5	Decimal	Linear Unit	
$\sigma_{y,max}$	Max. standard deviation from centre of any cluster in y direction	Maximum Distance	Fixes the maximum height of any cluster	Required	0.5	Decimal	Linear Unit	
K	Number of clusters	Goal	Determines number of clusters	Optional	0	Integer		
n	Number of iterations	Speed, Time		Optional	0	Integer		

Table 4 Example description of an algorithm (ISODATA)

The challenge is to define a common set of parameters that is neither too restrictive nor too broad. Their values should be derived using the user requirements and the source data. The value of any parameter defined by an algorithm would be a function of one or more of the common parameters. The relationship need not be a continuous function, it could be, for example, that for a range of values for a common parameter the algorithm's parameter could take a single value.

Since the use case focuses on the clustering of road accidents, the initial investigations will be based on clustering algorithms. There are a number of clustering algorithms but two, ISODATA and K-means, both include all source points in their generated clusters. K-means has a single parameter K which represents the number of clusters to be generated. As can be seen from Table 4 this parameter is shared with ISODATA. However, for ISODATA, K is only an optional parameter and is used with n , the number of iterations (optional), to put a stop on the iterative process. Its key parameters, $\sigma_{x,max}$ and $\sigma_{y,max}$, define the maximum size of the resulting clusters.

Further work will be carried out to examine how the output of each algorithm varies with their respective parameters in an attempt to define one or more common parameters that is defined by both the source data (for example, the initial point density) and the required output (target scale).

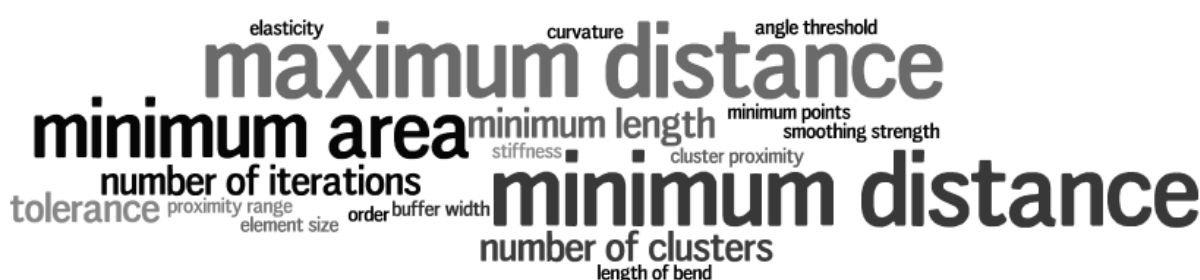


Figure 2 Word cloud of parameter names

5. Conclusions and further work

The next stage is to implement and formalise the algorithm classification described above by using it to semantically describe the generalisation services. This will be aided by a formalisation of the classification used by Li (2007).

The concept of a *semantic enablement layer* (as described by Janowicz et al., 2010) to facilitate service selection will be investigated further with a view to describing the algorithms/services using an ontology such as OWL-S (Martin et al., 2007) and a reasoning engine such as *Racerpro* to perform ontological matching (Lemmens et al, 2007; Fitzner et al., 2011; Janowicz et al, 2010).

Regarding the parameterisation of the algorithms, the survey has shown the diversity in their parameters emphasising the need to define a common set and work will continue to that aim.

References

- Douglas, D. & Peucker, T., 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature, *Cartographica*, 10 (2) pp. 112-122.
- Fitzner, D., et al., 2011. Functional description of geoprocessing services as conjunctive datalog queries. *Geoinformatica*, 15 (1) pp. 191-221.
- Foerster, T., et al., 2007. *A Generic Approach to Simplification of Geodata for Mobile Applications*. 10th AGILE International Conference on Geographic Information Science. Aalborg University, Denmark.
- Janowicz, K. et al., 2010. Semantic Enablement for Spatial Data Infrastructures. *Transactions in GIS*, 14 (2) p. 111.
- Lemmens, R., et al., 2007. Enhancing Geo-Service Chaining through Deep Service Descriptions. *Transactions in GIS*, 11 (6) pp. 849-871.
- Li, Z., 2007. *Algorithmic foundation of multi-scale spatial representation*. CRC Press.
- Maue, P., et al., 2009. *Semantic annotations in OGC standards* [online] Available at: <http://www.opengeospatial.org/standards/dp> [Accessed 14 January 2012]
- Martin, D., et al., 2007. Bringing Semantics to Web Services with OWL-S. *World Wide Web*, 10 (3) pp. 243-277.
- Open Geospatial Consortium, Web Processing Service. [online] Available at: <http://www.opengeospatial.org/standards/wps> [Accessed 14 January 2012]
- Revell, P., 2004. *Building on Past Achievements: Generalising OS MasterMap Rural Buildings to 1:50,000*. 7th ICA Workshop on Generalisation and Multiple Representation. Leicester, UK.
- STI Innsbruck, 2011. Envision - ENVironmental Services Infrastructure with ONtologies. [online] <http://www.envision-project.eu/> [Accessed 14 January 2012]
- Töpfer, F., Pillewizer, W. The Principles of Selection, *The Cartographic Journal*, 3 (1) pp. 10-16.
- Touya, G., et al., 2010. *Collaborative generalisation: formalisation of generalisation knowledge to orchestrate different cartographic generalisation processes*. Proceedings of the 6th international conference on Geographic information science. Zurich, Switzerland, Springer-Verlag.
- Visvalingam, M. and Whyatt, J. D., 1993. Line generalisation by repeated elimination of points. *The Cartographic Journal*, 30 (1) pp. 46-51.
- Ware, J. M., et al., 2003. Automated map generalization with multiple operators: a simulated annealing approach. *International Journal of Geographical Information Science*, 17 (8) pp. 743-769.
- Yan, H. and Weibel, R., 2008. An algorithm for cluster generalization based on the Voronoi diagram. *Computers & Geosciences*, 34 (8) pp. 939-954.

Appendix C

Paper presented to the 15th Workshop of the ICA Commission on Generalisation and Multiple Representation. Istanbul, Turkey, 13th-15th September 2012.

An Ontological approach to On-demand Mapping

Nick Gould¹, Omair Chaudhry²

¹Manchester Metropolitan University, Manchester, UK
Email: nicholas.m.gould@stu.mmu.ac.uk

² Manchester Metropolitan University, Manchester, UK
Email: o.chaudhry@mmu.ac.uk

Introduction

Automatic generalisation for map production has been in use for decades (Li, 2007a). The process is still, however, only semi-automatic in that the expert selects and sequences the required generalisation operators and the algorithms that implement them and provides parameter values. Different techniques can be applied to rural and urban areas at the discretion of the expert, working to a fixed target scale and with familiar feature types (Regnauld and Revell, 2007). But in the case of on-demand mapping the expert will be replaced by a system that will be able to automatically select, sequence and execute map generalisation operations according to user requirements.

The aim of this project is to develop a workflow generation engine that is at the core of an on-demand mapping system (Balley and Regnauld, 2011). The concept of *abstract* tasks as represented by generalisation operators (*Simplification*, *Amalgamation* etc.) and *concrete* tasks as represented by algorithms, that implement operators, will be employed. The separation of abstract and concrete tasks allows for a separation of the definition of the requirements and its implementation (by web services). The process to generate a workflow for on-demand mapping can be broken down as follows:

1. Define abstract tasks - operators
2. Define concrete tasks - algorithms
3. Generate workflow
4. Execute workflow

Before we can automate any task it is necessary to understand it (Georgakopoulos, et al., 1995). We need to formalise the *why*, *when* and *how* of generalisation (McMaster and Shea, 1992). This is particularly important if we want an open, interoperable system. Ontologies allow us to semantically enrich the descriptions of both data and services such that the data and services can become machine-interpretable (Lutz, 2007). This paper focuses on the semantic description of generalisation operations and algorithms using ontologies.

To work effectively, an ontology has to be designed for a specific task (Noy and McGuinness, 2001). Section 3 describes the development of an ontology for automatically selecting generalisation operators. Section 4 describes an ontology for the automatic selection of algorithms to implement the chosen operators. Possible options for deploying the ontologies are discussed in section 5 along with some conclusions. The next section discusses previous work in geospatial ontologies and in on-demand mapping.

Related work

One possible solution to on-demand mapping is to avoid the dynamic generalisation of data and use a Multi Resolution DataBase (MRDB) (Dunkars, 2004). However, if we define on-demand mapping as generalisation according to user requirements, and potentially integrating user-supplied data, then the MRDB approach is not applicable. Bernier and Bédard (2007) describe a hybrid approach – if the data can be generalised quickly and without human intervention then it should be – otherwise the data should be extracted from the MRDB.

If an on-demand system is to integrate user-supplied data in an ad-hoc fashion then automatic, on-demand, generalisation is required. However, if the process is to be completely automated then we first need to formalise the knowledge required to produce a generalised map (Touya et al, 2010). Generalisation is achieved by applying one or more transformations or *operators* (Sarjakoski, 2007). However, following a series of interviews with cartographers, Rieger and Coulson (1992) concluded that there was no consensus over the description of these operators; cartographers had different definitions of the same term and different terms for the same definition. Rieger and Coulson were attempting to elicit *declarative* knowledge about the procedures as opposed to *procedural* knowledge, which describes how the task is carried out. Declarative knowledge, that knowledge contained in declarations about the world, can be extended by reasoning processes that derive additional knowledge (Genesereth and Nilsson, 1998). Can such a method be applied to generalisation?

There have been a number of attempts to classify and describe generalisation operators (Foerster, et al., 2007a; McMaster and Shea, 1992; Roth, et al., 2011) but the problems highlighted by Rieger and Coulson (1992) remain. As well as differences between the proposed categories of operators there are also differences in naming (*Aggregation* or *Combine*?) and in granularity; McMaster and Shea (1992) define *Smoothing*, *Enhancement* and *Exaggeration* where Foerster et al. (2007a) simply define *Enhancement*. There is also disagreement as to what functions can be regarded as generalisation operators. For example, is *Symbolisation* a generalisation operator (McMaster and Shea, 1992; Roth et al., 2011) or a pre-processing step (Foerster et al., 2007a)?

The use of different operator taxonomies in closed systems does not matter, but, if we are to develop an interoperable on-demand system, an agreed taxonomy *and* the semantic description of the operators is required. This is because we cannot simply ask for a web service that performs *Smoothing*, say, since that operation can be performed by a number of different algorithms (Gaussian, Cubic Spline, Fourier transform etc.), often with different results. Similarly, some operators apply to different geometry types and will need to be implemented by different algorithms. Likewise some algorithms specialise in different feature types e.g. buildings (Guercke and Sester, 2011). Thus these details need to be formally defined so that automatic selection and execution is possible by the on-demand system.

Li's study (2007b) of generalisation *algorithms* (rather than operators) provides a possible framework for the semantic description of the generalisation process. He focuses on algorithms and groups them by geometry and by what function they perform; point reduction of areas, for example.

At some stage in the development of an on-demand mapping system there will be a need for Knowledge Acquisition (Kilpeläinen, 2000; Mustiere, 2005; Rieger and Coulson, 1993) but first it is necessary to define the type of knowledge that needs to be acquired and how it is to be encapsulated. The dominant methods for encapsulating cartographic knowledge are *rules* and *constraints*. The rule-based approach involves defining a set of condition-action pairs that will solve particular problems (Sarjakoski, 2007). Rules encapsulate procedural knowledge. The disadvantage of this approach is that a rule has to exist for every eventuality

which means a large number of rules need to be defined for a viable system (Harrie and Weibel, 2007). Unlike rules, constraints do not prescribe how a problem should be solved only the condition that should be maintained (Neun et al., 2009).

Formalisation of knowledge can lead to the discovery of new knowledge as long as appropriate formalisation tools are available (Kilpeläinen, 2000). One such tool is the ontology - the explicit specification of the objects, concepts and the relationships in a body of knowledge concerning a particular subject or domain (Gruber, 1993). Ontologies have the advantage of allowing the sharing and reuse of formalised knowledge (Gruber, 1993). The semantic description of geospatial operations, and the web services that implement them, using ontologies to allow for automatic selection is not new (Klusch et al., 2005; Lutz, 2007; Lemmens, et al., 2007) but there has been little focus on the particular problems of generalisation. Touya et al. (2011) have made progress on a generalisation ontology but not specifically for on-demand mapping.

The next section describes the process for developing a generalisation operator ontology.

Developing the operator ontology

There is no single, ideal, methodology for designing an ontology (Noy and McGuinness, 2001). The authors' first attempt to develop a generalisation ontology for on-demand mapping involved attempting to capture, in one-step, all the knowledge that could be used to describe the generalisation process. This led to a large, cumbersome, and ultimately unusable ontology. An alternative approach was taken, that involved defining an ontology for a specific purpose.

The purpose of the *operator* ontology is to describe the properties, behaviours and relations of generalisation operators in such a way that they can be selected automatically. The ontology will be designed by reference to a road accident use case (Figure 1). The model will then be tested against further use cases such as the cycle route planner described by Balley and Regnauld (2011). The requirement of the user is to view the road accidents at a detailed level, where no generalisation is required – showing the road network as polygons and individual accidents (Figure 1a) – and at a small, city-wide scale.



Figure 1 Road accident use case

The aim of the system is to produce a map that *maintains legibility* as the scale is reduced (*Why generalise*). We can decide when to generalise by describing a number of geometric conditions (McMaster and Shea, 1992). For example, the road network which is described using an area geometry (Figure 1a) becomes *congested* at a smaller scale (Figure 1b) and also suffers from *imperceptibility* as the lines that define the road boundaries become too close. The accident dataset at a smaller scale (Figure 1c, shown separately) suffers from *congestion*, *coalescence* and *overlap*. We also have to define a number of measures, such as feature

density, to evaluate when a condition has been reached (Stigmar and Harrie, 2011). We can then say that generalisation is required *when* a particular geometric condition occurs. The condition is resolved by one or more operators (*How* to generalise).

Rather than simply present the completed ontology we have described below the decisions and steps taken to build the ontology. This will facilitate criticism of the resultant ontology and help inform further development. This was thought to be particularly important for ontologies that describe a process rather than a set of tangible objects.

The first version of the ontology can be seen in Figure 2. The labelled solid lines represent object properties and the unlabelled dotted lines represent “is-a” sub-class relationships.

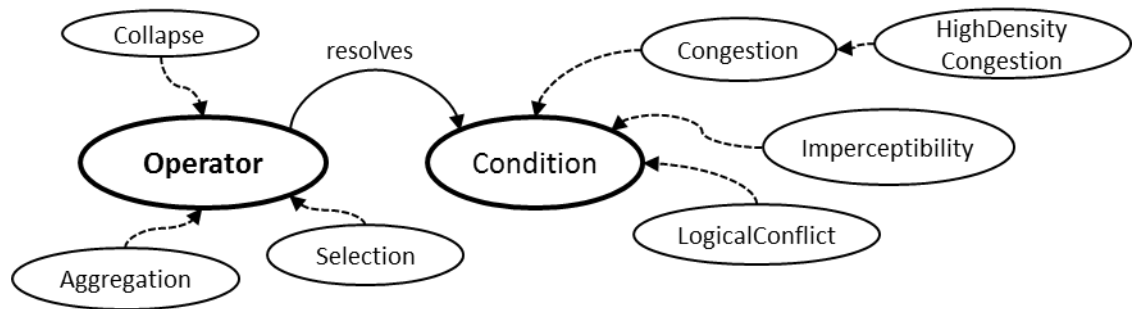


Figure 2 Operator ontology - version 1

The *LogicalConflict* condition is a renaming of the *Conflict* condition defined by McMaster and Shea (1992). Such a condition may occur, for example, when a number of accidents are displayed but the road they lie on has been eliminated for some reason.

Operators can be added to the ontology and linked to one or more conditions (e.g. *Collapse* resolves *HighDensityCongestion*). The measure for *HighDensityCongestion* can be modelled by creating a data property *hasDensity* and adding it to the *HighDensityCongestion* condition with a threshold value. This will need refinement since we will likely have different measures for the congestion of different geometries. The ontology was implemented in Protégé (Horridge, 2011) which allows for the querying of an ontology. So the query:

Operator and *resolves* some **HighDensityCongestion**

might return a number of operators. However, not all operators work on the same geometry types. For example, *Amalgamation* applies to area features and not point features; *Collapse* can apply to areas and lines but not points. By introducing a geometry class and linking specific operators to specific geometry classes, we can reduce the number of operators applicable to a given situation, thus facilitating the automatic selection of an operator. The refined version of the ontology can be seen in Figure 3.

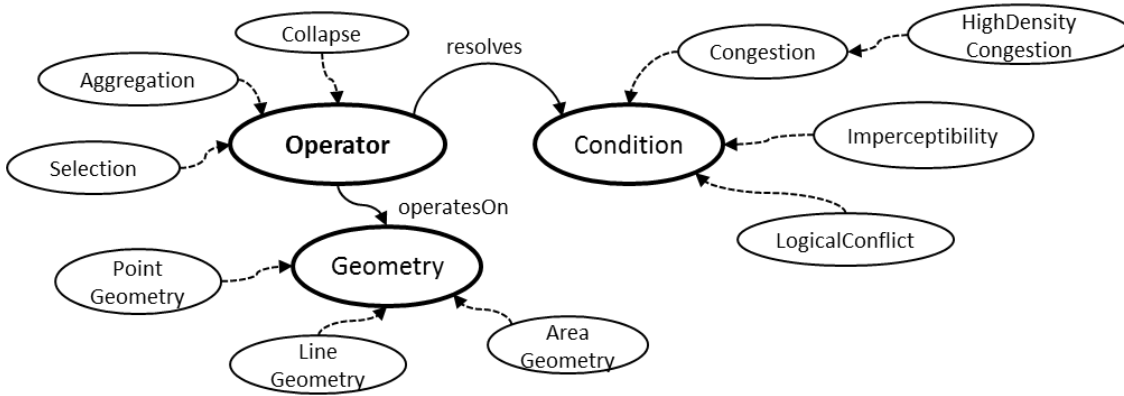


Figure 3 Operator ontology - version 2 (some classes and relations omitted for clarity)

The query can be refined:

Operator and *resolves* some **HighDensityCongestion** and *operatesOn* some **PointGeometry**

Only the operators *Aggregation* and *Selection* would be returned since they are the only two operators that were defined as resolving congestion specifically in point features. The ontology can be further refined when we consider the *Selection* operator in more detail. *Selection* can be used in our use case to reduce congestion by only selecting the most serious road accidents or the most important roads. However, for *Selection* to work the dataset needs an attribute that can be used to rank its features. The ontology therefore needs a concept of a dataset, in particular a ranked dataset, and the concept of an operator transforming a dataset (Figure 4).

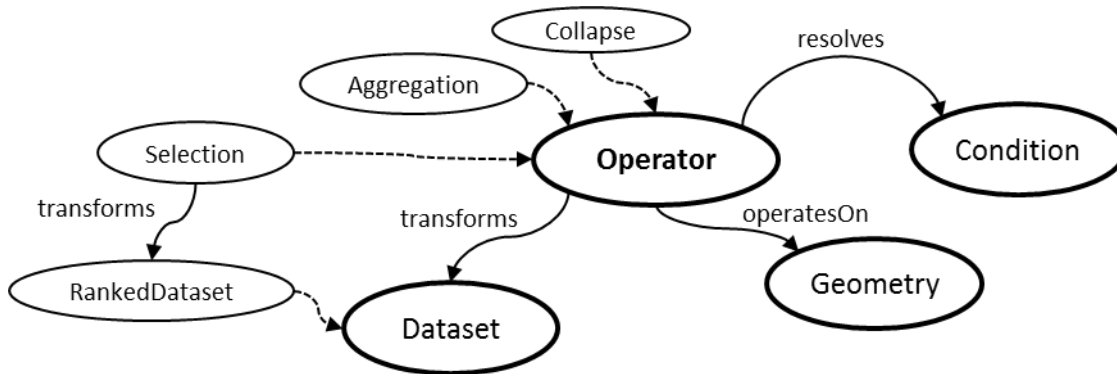


Figure 4 Operator ontology - version 3 (some sub-classes omitted for clarity)

It can be seen that there are a number of relations and classes that could have been defined but were not; for example the relationship between a dataset and its geometry or the possible sub-classes of a dataset (Road dataset, Accident dataset, for example). However, the ontology has been designed on the principle of defining only that which is necessary to fulfil the defined aim (Noy and McGuinness, 2001).

A number of measures were defined for the two datasets in the use case. For point data (the accidents) a density measure of congestion was utilised, based on the number of points per unit map area (pixels). For polygon data (the road network) two measures were defined; an average polygon width (in pixels) as a measure of imperceptibility, on the understanding that if the road section is too narrow then it will be difficult for the viewer to distinguish between opposite sides of the road section. The second measure was for congestion and uses a total

feature area per unit map area. The measures were implemented using the Geotools JAVA library and tested on sample data from Greater Manchester. Arbitrary threshold values for the measures were defined and used to indicate whether generalisation was required. The aim is that once the conditions have been identified then the ontology can be queried to determine the appropriate operators to resolve the conditions. This application of measures to trigger generalisation requires further refinement. For the point density measure the effect of symbol size was ignored and no account was made for the spatial distribution of features in either dataset. In addition, each dataset was considered in isolation. It is unlikely that there is a single measure for a condition that is appropriate in all cases and a combination of measures might need to be applied (Mustiere, 2005; Stigmar and Harrie, 2011).

The ontology itself is not complete and there are some unanswered questions. For example, should the operator ontology include the concept of *feature type*? Also, *Amalgamation* may be identified as a suitable operator for congested area features, which may be appropriate for buildings but not for roads or a river network.

The ontology also lacks the concept of precedence. If a query returns two operators that meet the conditions then does this mean that both operators should be applied to the dataset? If so, then in what order? If we apply the first operator and the condition persists do we try the second operator or repeat the first but with a different parameter value? Such a question may lie outside the remit of the ontology and be the responsibility of a Problem Solving Method (PSM) (Gómez Pérez and Benjamins, 1999), which is required to manage the process of constructing and asking the queries and then acting on the results. For example, an agent-based or other optimisation method may be used to define the ideal sequence of proposed algorithms.

Kilpeläinen (2000) refers to the knowledge that is used to select the right generalisation operator for the task as *procedural* knowledge. For the ontological approach to be effective, rather than having to explicitly state the procedural knowledge in the form “operator X resolves condition Y”, the procedural knowledge could be derived from the declarative knowledge by reasoning (Genesereth and Nilsson, 1998). In effect, the “operator resolves condition” relation needs to be made redundant by describing both conditions and operators in such a way that we can derive the relation by query. This requires a more explicit statement of what the operators do and what the conditions are.

The next stage is to develop an ontology that will help select algorithms to implement the selected operators. A separate algorithm is required since algorithms have a different set of properties from operators and an operator can be implemented by a number of algorithms.

Developing the algorithm ontology

Before developing the algorithm ontology a survey of generalisation algorithms was done with the intention of informing the design process by highlighting the attributes and behaviours of generalisation algorithms. Algorithms for point aggregation, line *Simplification*, line *Smoothing*, and building *Amalgamation* were examined and common attributes documented (Gould, 2012).

In addition to the operators they implemented and the geometry types they applied to, algorithms varied by *feature type* - some algorithms were specific for roads or buildings, for example - and by *terrain* - some algorithms were targeted at mountain roads or rural buildings. Algorithm *parameters* provide further variety - algorithms performing the same task, e.g. line simplification may have different parameters. *Scale* also provides an additional

layer of complexity; some algorithms are designed for specific source and target scales. This explains why we need a separate algorithm ontology.

A similar incremental approach to the operator ontology design was used to answer the question: how do we describe an algorithm so that it can be automatically selected? An initial version of the ontology can be seen in Figure 5.

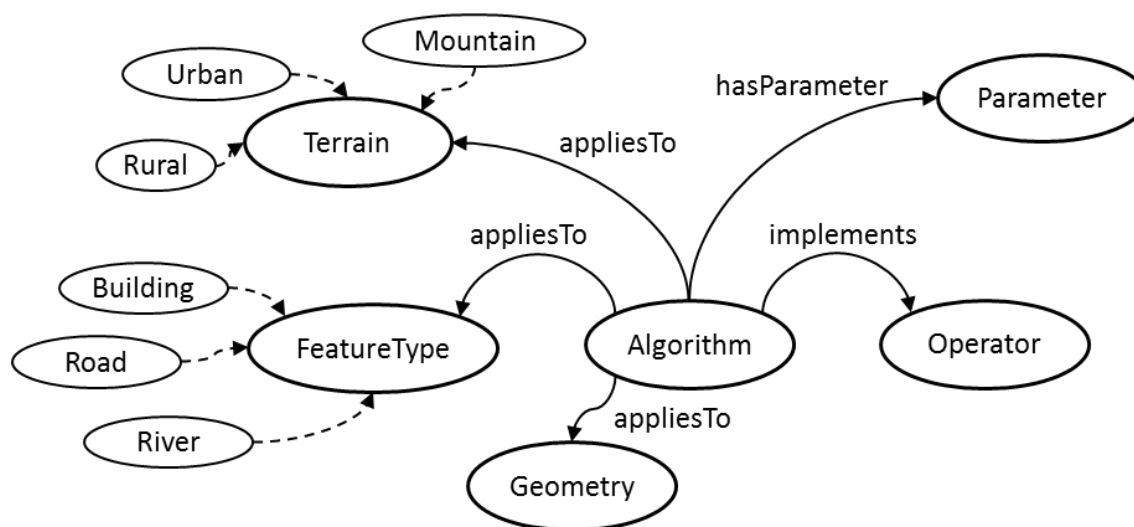


Figure 5 Algorithm ontology (some sub-classes omitted for clarity)

An example query, that aims to find an algorithm for road smoothing might be:

Algorithm and *implements* some **Smoothing** and *appliesTo* some **RoadFeatureType**

In practice an algorithm can be regarded as an abstraction for a *web service*. It is not practical, or necessary, to have a model where we search for a service that implements a particular algorithm. It is unlikely that any service could advertise itself only by the algorithm it implements.

As with the operator ontology, the algorithm ontology requires further refinement. Algorithms that implement multiple operators, such as line simplification *and* smoothing, need to be modelled. Further consideration of which concept should sit within which ontology may be necessary. For example, should the ‘terrain’ concept sit within the operator ontology?

Conclusions and further work

We believe that although there are still questions to be answered, the ontological approach to on-demand mapping merits further investigation. But, to what extent can we use ontological reasoning to develop a workflow for on-demand mapping? Is the ontological approach merely a stepping stone to help inform another approach or is it an end in itself? How could the ontologies be applied?

The standard for implementing geospatial web services is the OGC’s Web Processing Service (WPS) protocol. However, the protocol does not provide for *semantic interoperability* (Janowicz et al., 2010); there is no method of adding machine readable descriptions to a service. One possible solution that will be investigated further is the Semantic Enablement Layer (Janowicz et al., 2010) where a *Web Ontology Service* injects semantics into both data and processing services. A *Web Reasoning Service* can then be used to match a processing

service to a dataset. Previous work on the generation of workflows for on-demand mapping from a set of tasks and task precedencies (Gould and Chaudhry, 2012) could be employed to generate valid workflows from the output of a Semantic Enablement Layer. The Web Ontology Service could be employed to maintain a shared set of on-demand mapping ontologies.

One major obstacle yet to be resolved is the problem of how to *automatically* provide parameter values to the selected services especially since any two algorithms performing the same generalisation operation may have different parameters. Even if the two algorithms had parameters with a common name such as *minimum distance*, their concept of what a minimum distance means may differ. One possible approach would be to define a common set of parameters to be used by all services, extending the work on line simplification ratios of Foerster et al. (2007b). It would then be the responsibility of any service implementing an algorithm to translate the common parameter values to local parameter values. Values for the common parameters could be derived from the geometric condition measures described in the Operator ontology. For example, a high value for a condition could lead to a correspondingly high value for a parameter for the selected algorithm.

Acknowledgements

This project is funded by the Dalton Research Institute at Manchester Metropolitan University and the Ordnance Survey of Great Britain. Thanks to Nico Regnauld and Sandrine Balley at the Ordnance Survey and Martin Stanton at MMU for their guidance and to the anonymous reviewers for their suggestions.

References

- BALLEY, S. and REGNAULD, N., 2011. *Models and standards for on-demand mapping*. 25th International Cartographic Conference. Paris.
- BERNIER, E. and BEDARD, Y., 2007. A Data Warehouse Strategy for on-Demand Multiscale Mapping. In: W. A. MACKANESS, et al. eds., *Generalisation of Geographic Information*, Amsterdam: Elsevier Science B.V., pp. 177-198.
- DUNKARS, M., 2004. *Automated Generalisation In A Multiple Representation Database*. 12th Int. Conf. on Geoinformatics - Geospatial Information Research: Bridging the Pacific and Atlantic. University of Gävle, Sweden.
- FOERSTER, T., et al., 2007a. *Towards a formal classification of generalization operators*. International Cartographic Conference 2007. Moscow.
- FOERSTER, T., STOTER, J., KÖBBEN, B. and VAN OOSTEROM, P. 2007b. *A Generic Approach to Simplification of Geodata for Mobile Applications*. 10th AGILE International Conference on Geographic Information Science. Aalborg University, Denmark.
- GEORGAKOPOULOS, D., et al., 1995. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, 3 (2) pp. 119-153.
- GENESERETH, M. R., and NILSSON, N. J., 1987. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers.
- GUERCKE, R. and SESTER, M., 2011. *Building Footprint Simplification Based on Hough Transform and Least Squares Adjustment*. ICA Workshop on generalisation. Paris, France.
- GOMEZ-PEREZ, A., BENJAMINS, R. 1999. Overview of Knowledge Sharing and Reuse Components. Ontologies and Problem Solving Methods. In: *Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods (KRR5)*, Stockholm.
- GOULD, N., 2012. *Semantic description of generalisation web services for on-demand mapping*. 1st AGILE PhD School, Wernigerode, Germany, Shaker Verlag, 13-14th March 2012.

- GOULD, N. and CHAUDHRY, O. Z., 2012. *Generation and validation of workflows for on-demand mapping*. Geoprocessing 2012. Valencia, Spain.
- GRUBER, T. (1993) A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2), pp. 199-220.
- HARRIE, L. and WEIBEL, R. 2007. Modelling the Overall Process of Generalisation. In: MACKANESS, W. A., RUAS, A. and SARJAKOSKI, L. T. (eds.) *Generalisation of Geographic Information*, Amsterdam: Elsevier Science B.V. pp. 67-87.
- HORRIDGE, M., 2011. *A Practical Guide to Building OWL Ontologies Using Protege 4 and CO-ODE Tools Edition 1.3*. [online] <http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/> Accessed 11th May 2012.
- JANOWICZ, K., et al., 2010. Semantic Enablement for Spatial Data Infrastructures. *Transactions in GIS*, 14 (2) p. 111.
- KILPELÄINEN, T. 2000. Knowledge Acquisition for Generalization Rules. *Cartography and Geographic Information Science*, 27(1) pp. 41-50.
- KLUSCH, M., et al., 2005. Semantic Web service composition planning with OWLS-Xplan. *AAAI Fall Symposium - Technical Report*, FS-05-01 pp. 55-62.
- LEMMENS, R., et al., 2007. Enhancing Geo-Service Chaining through Deep Service Descriptions. *Transactions in GIS*, 11 (6) pp. 849-871.
- LI, Z., 2007a. Digital map generalization at the age of enlightenment: A review of the first forty years. *The Cartographic Journal*, 44 (1) pp. 80-93.
- LI, Z., 2007b. *Algorithmic foundation of multi-scale spatial representation*. CRC Press.
- LUTZ, M., 2007. Ontology-Based Descriptions for Semantic Discovery and Composition of Geoprocessing Services. *GeoInformatica*, 11 (1) pp. 1-36.
- MCMASTER, R. B. and SHEA, K. S., 1992. *Generalization in digital cartography*. Washington, D.C.: Association of American Geographers.
- MUSTIERE, S., 2005. Cartographic generalization of roads in a local and adaptive approach: A knowledge acquisition problem. *International Journal of Geographical Information Science*, 19 (8-9) pp. 937-955.
- NEUN, M., BURGHARDT, D. and WEIBEL, R. 2009. Automated processing for map generalization using web services. *GeoInformatica*, 13 (4) pp. 425-452.
- NOY, N. F. and MCGUINNESS, D., 2001. *Ontology Development 101: A Guide to Creating Your First Ontology*. Stanford University.
- REGNAULD, N. and REVELL, P., 2007. Automatic Amalgamation of Buildings for Producing Ordnance Survey 1:50 000 Scale Maps. *The Cartographic Journal*, 44 (3) pp. 239-250.
- RIEGER, M.K. and COULSON, M.R.C., 1992. Consensus or Confusion: Cartographers' Knowledge of Generalization. *Cartographica*, 30 (2 & 3) pp. 69-80.
- ROTH, R., et al., 2011. A typology of operators for maintaining legible map designs at multiple scales. *Cartographic Perspectives*, 68.
- SARJAKOSKI, L. T. 2007. Conceptual Models of Generalisation and Multiple Representation. In: MACKANESS, W. A., RUAS, A. and SARJAKOSKI, L. T. (eds.) *Generalisation of Geographic Information*, Amsterdam: Elsevier Science B.V., pp. 11-35.
- STIGMAR, H. and HARRIE, L., 2011. Evaluation of Analytical Measures of Map Legibility. *The Cartographic Journal*, 48 (1) pp. 41-53.
- TOUYA, G., et al., 2010. *Collaborative generalisation: formalisation of generalisation knowledge to orchestrate different cartographic generalisation processes*. Proceedings of the 6th International Conference on Geographic Information Science. Zurich, Switzerland, Springer-Verlag.

Appendix D

Paper presented to the 16th Workshop of the ICA Commission on Generalisation and Multiple Representation, Dresden, Germany, 23rd – 24th August 2013.

A prototype for ontology driven on-demand mapping of urban traffic accidents

Nicholas Gould¹, Jianquan Cheng²

Manchester Metropolitan University, Chester Street, Manchester, UK

Email: ¹nicholas.m.gould@stu.mmu.ac.uk

Email: ²j.cheng@mmu.ac.uk

Can the concepts of cartographic generalisation be formalised in an ontology with sufficient detail to allow the process to be automated?

1. Introduction

Government, both national and local, is making increasing amounts of spatial data freely available. The DataGM website, for example, provides access to georeferenced data for road traffic accidents, fire and rescue incidents, bus stops, bus routes and traffic signals in Greater Manchester (Trafford Council 2012). However, how can thousands of road accidents be mapped legibly by the non-expert cartographer without obscuring the underlying road network? Tools such as the Google maps API provide only a partial solution in that they merely overlay data on base maps. There is no integration of user-supplied data. What is required is cartographic generalisation on-demand. But to automate the map creation process it is necessary to formalise the knowledge required for generalisation (Touya et al. 2010).

2. Why use an ontology?

The prevailing paradigm for automatic generalisation is constraint-based, in particular agent-based, modelling (Harrie and Weibel 2007). Agent-based systems require a knowledge base that has to be updated each time a new generalisation algorithm is introduced or when user requirements change (Taillandier and Taillandier 2012). What happens when an end-user wishes to map features of an unfamiliar type, such as road accidents? The knowledge required to generalise these features, their attributes, relevant operations and relations with other features, has to be encoded. In effect, cartographic knowledge is embedded in the configurations of sophisticated software applications. Ideally that knowledge, once defined, would be shared.

One possible option for representing domain knowledge in a sharable and reusable manner is to employ ontologies (Gruber 1993). An ontology captures the semantics of the concepts in a domain and is not merely a classification (Kavouras and Kokla 2008). It has been argued that all information systems are based on implicit ontologies and making the ontology explicit avoids conflicts between ontological concepts and their implementation (Fonseca et al. 2002). The use of ontologies to realise semantic interoperability in a distributed environment is well researched (Lemmens et al. 2007; Lüscher et al. 2007; Janowicz et al. 2010) and Regnauld (2007) proposed an on-demand mapping system based on ontologies.

Ontologies are more than taxonomies; we can reason with them and apply them to decision-making. Such ontologies, used in scientific workflows, are rare (Janowicz et al. 2012). In other domains ontologies have been used to match students to courses (Kontopoulos et al. 2008) and applicants to jobs (García-Sánchez et al. 2006). This paper describes an attempt to use an ontology to model the process of generalisation, in an effort to facilitate the automation of map generation at different scales.

The aim of this project is to determine the *why*, *when* and *how* of generalisation (McMaster and Shea 1992). The need to produce a legible map is the reason *why* we need to generalise.

The existence of geometric conditions (congestion, imperceptibility) in the mapped data determines the *when*. The existence of these conditions can be determined by using measures such as the density and distribution of features (Stigmar and Harrie 2011). *How* is answered by the use of generalisation operations such as Amalgamation and Collapse. The goal is to use an ontology to help choose which measures and operations to apply.

There is no single correct way of modelling a domain and ontological engineering is necessarily an iterative process (Noy and McGuinness 2001). There are a number of methodologies available to guide the process (Sure et al. 2009) but in our case the “simple” method described by Noy and McGuinness (2001) was employed. This involves defining a set of *competency questions* that the ontology is expected to answer. These include: *what measure algorithm should be used for a particular condition? What operation will alleviate the condition?*

The ontology employs a (loose) medical analogy which describes *conditions* (such as feature congestion) that are characterised by *symptoms* (e.g. high feature density) which have *remedies* (e.g. feature count reduction). The remedies are implemented by operations which in turn are implemented by transformation¹⁷ algorithms.

The applicability of an operation to a given condition of a given set of features is governed by a number of factors, primarily geometry (for example, point features cannot be collapsed; pruning only applies to line features, specifically a network of line features). Geometry alone is not sufficient, however. The choice of operation is also governed by a number of *requirements* and *restrictions*. SelectionByAttribute requires that the source data has an attribute (field) that can be used to rank the features by importance. Amalgamation is restricted from application to a road network since Amalgamation is a form of Abstraction, which is forbidden for a Network by the ontology.

The intention is to describe the operations sufficiently that they can be selected automatically. It is also necessary to define the output geometry of the operation since that will influence the selection of subsequent operations. Any unintended consequences of the operation also have to be described. For example the process of Amalgamation, in abstracting the original features, will remove any importance attribute from the features and thus prohibit the use of SelectionByAttribute in subsequent operations.

The ontology, stored as an OWL2 (Web Ontology Language) file, was developed using the Protégé ontology editor (Horridge 2011) and it could be tested by issuing queries from within the editor. However, to fully test the concept a prototype was developed.

3. Implementation

The prototype consists of using the ontology to select appropriate measure algorithms, applying those measure algorithms and, if a condition was identified, selecting an operation (and transformation algorithm) to remedy the condition.

In the current prototype (Figure 1) the measure and transformation algorithms, implemented as Java methods, form part of the system, but it is envisaged that these will be provided in future by web services such as WEBGEN (Burghardt et al. 2005). The following sections describe the main components of the prototype.

3.1 Source data

In the first instance the prototype is restricted to mapping traffic accidents with roads as a base. The accidents are in Greater Manchester, UK, over a 12 year period. Each accident has a

¹⁷ The word transformation is used instead of generalisation because of doubts over what operations actually constitute generalisation (Foerster et al. 2007).

severity attribute with values of 1 (fatal), 2 (severe), or 3 (slight). The base road network is Ordnance Survey's MasterMap road features (polygons) and their Integrated Transport Network (ITN) road network (lines)¹⁸.

The source dataset is described as an *individual* (or instance) in the ontology using attributes such as Geometry and FeatureType. In the current implementation the source data is held as Shape files but future versions could use web services.

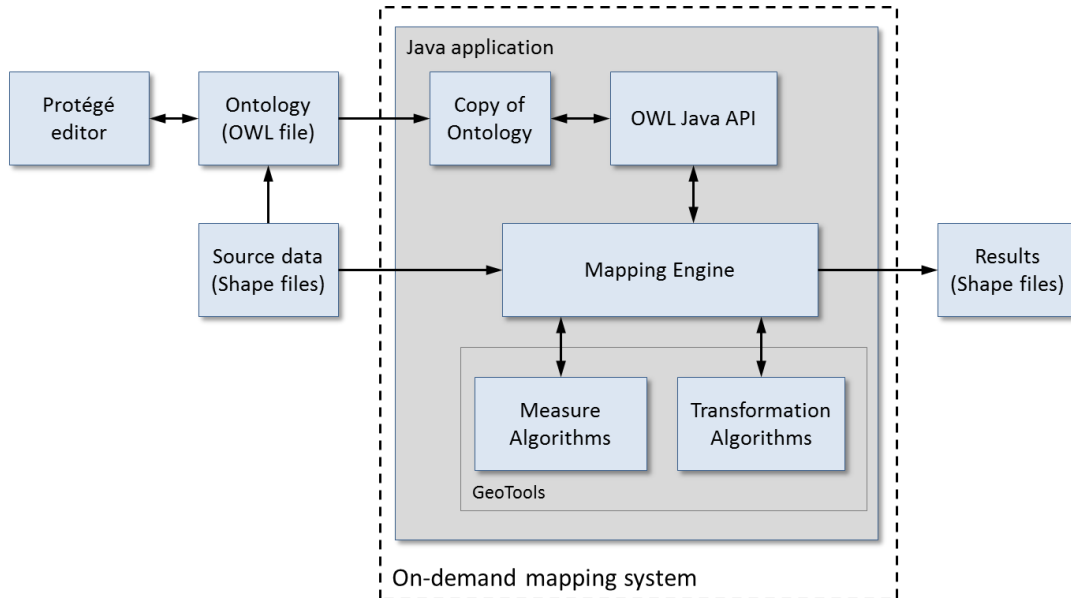


Figure 1. On-demand mapping system prototype architecture.

3.2 Using the ontology

Interaction between the Java application and the ontology is done via the OWL API (Horridge and Bechhofer 2011). Most of the calls to the ontology consist of queries in the form of Manchester OWL syntax. The syntax is verbose but human-readable, which makes development easier. The following is an example of query string generated by the Mapping Engine and executed against the ontology, which will return a list of measure algorithms meeting the specified criteria:

```
MeasureAlgorithm and measures some HighFeatureDensity and
hasInputGeometry some AreaGeometry
```

The first usage of the ontology is to identify an appropriate measure algorithm for each *mapped feature collection*, where a mapped feature collection is defined as a set of features, of the same type, in the user's selected bounding box.

3.3 Measure algorithms

A number of measure algorithms were developed for the prototype, including one to measure the density of point features and others to measure the density of road features (as areas and as lines). There is also an algorithm to measure the density of generic area features, such as amalgamated clusters of accidents. The algorithms begin by identifying clusters in the source data. What constitutes a cluster is determined by scale. If the density of features in a cluster exceeds a threshold then the clusters are returned flagged as high density (Figure 2). The

¹⁸ The ITN network is used to simulate a collapse of the Master Map road features.

focus on the density of features is because we are primarily interested in resolving the congestion of features.

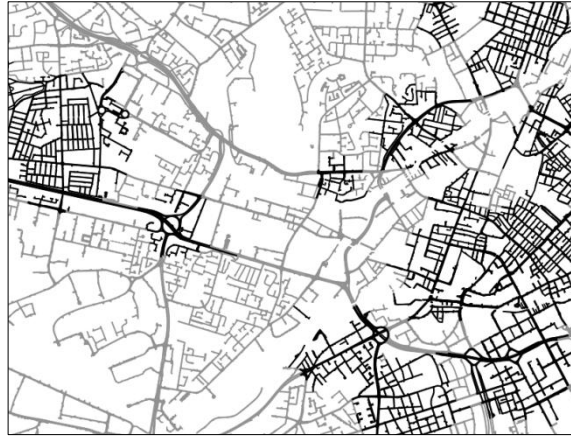


Figure 2. Regions of high crossroad density (in black).

If a mapped feature collection is deemed to have a particular condition (e.g. congestion) then the next stage is to select an appropriate operation to remedy the problem and then to choose an algorithm that implements that operation.

3.4 Transformation algorithms

The transformation algorithms implemented in the prototype are governed by a *DegreeOfGeneralisation* parameter (Zhou and Jones 2003). The value of this parameter (1 = low, 9 = high) is governed by the output from the respective measure algorithm. The higher the number of congested features found, the higher the value of the *DegreeOfGeneralisation*. The exception is the Collapse algorithm which is a binary - all or nothing - operation. A more sophisticated version of the prototype would consider then number of clusters and their distribution rather than simply the total number of clustered features.

The *SelectionByAttribute* operation will only be suggested by the ontology if the source data has an *ImportanceAttribute* defined in its ontology description. This is simply the name of a data attribute (field) that can be used to rank the features by importance. The accident data has a *severity* attribute that can serve as such.

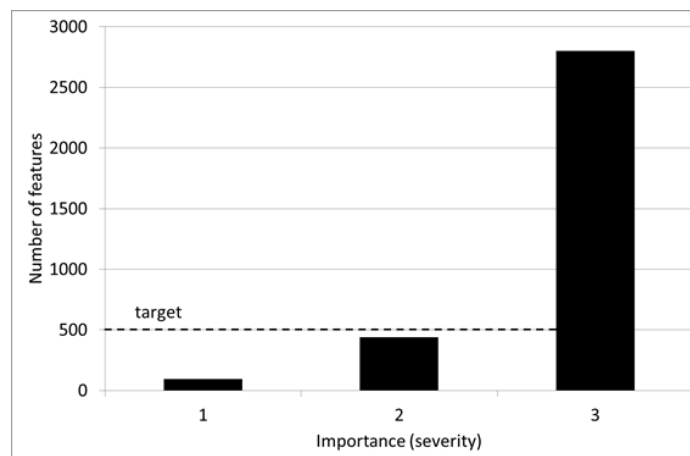


Figure 3. Distribution of accident importance

The algorithm that implements *SelectionByAttribute* uses the *DegreeOfGeneralisation* to determine the number of features to retain. This value is then used to determine which features to retain. For example, if the number of features to retain was 500 then the algorithm would return features of importance value of 1 and 2 (Figure 3).

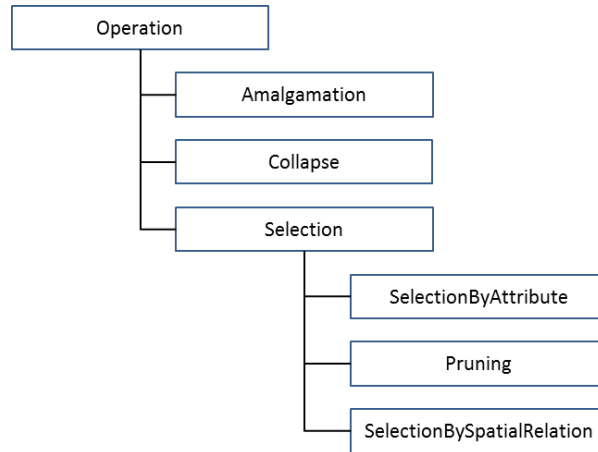


Figure 4. Operation sub-classes in the ontology

The same principle is used to govern the pruning algorithm applied to the road network. *Pruning*, like *SelectionByAttribute*, is a form of *Selection* (Figure 4). However, in this case the algorithm uses the *DegreeOfGeneralisation* to determine the *total length* of the features to be retained. The pruning algorithm uses strokes to determine which road features to retain (Thomson and Richardson 1999). The longer the stroke the higher its importance and the more likely it is to be retained.

3.5 Workflow

The user selects two data sources (accidents and roads). The order of selection is important; it means that the roads have to be generalised with respect to the accidents. The user is presented with a starting bounding box with the features displayed at a large scale. The workflow (Figure 5) is triggered when the user either pans or changes scale by zooming. The features in the bounding box are defined as a *mapped feature collection*, one for each feature type (i.e. one for road sections and one for accidents).

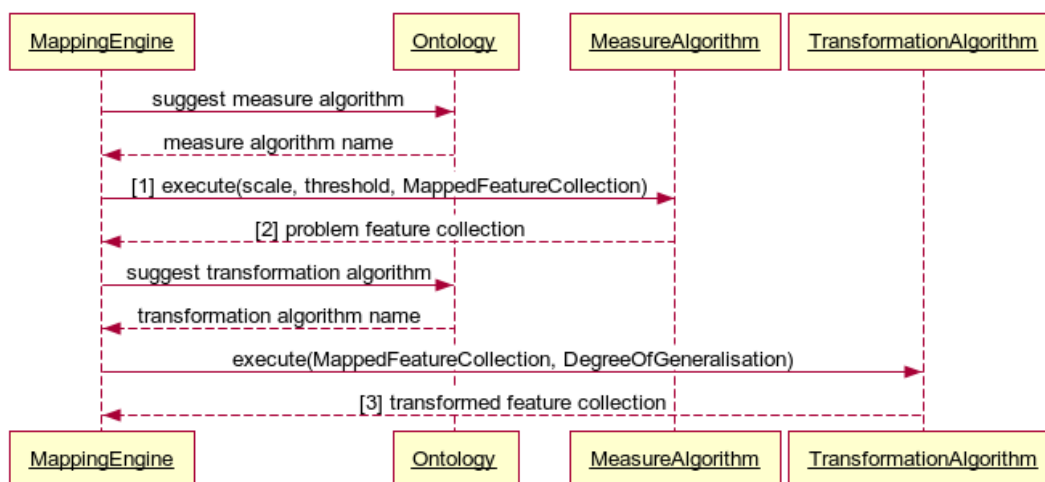


Figure 5. UML sequence diagram (single mapped feature collection)

The workflow is applied to each mapped feature collection in turn for a given scale. The entire process is complete when there are no *conditions* identified by the measure algorithms for any of the mapped feature collections. If the user chooses another scale then the process is repeated, starting with the original source data.

If the *problem feature collection* (those features in the mapped feature collection identified as having the condition) is empty (step [2], Figure 5) then the sequence stops for that particular mapped feature collection. After step [3], step [1] is called again to assess the effect of the transformation. As depicted, the transformation algorithm is applied to all features in the mapped feature collection, not only those identified as problem features. This is, in fact, dependent on the transformation operation. For example, for *collapse* the transformation will apply to all features in the class but for *amalgamation* only those features in the problem feature collection will be affected.

The application of a transformation algorithm will change the data (e.g. from a cluster of point features to an area feature). However, the changes enacted by the transformation have to be reflected in the ontology; this is why a working copy of the ontology is made (Figure 1). The semantics of the features may have changed as well as the geometry. For example, the feature type of a set of amalgamated accidents is no longer AccidentFeatureType. The changes in the semantics may well effect what measure and transformation algorithms are applicable in subsequent iterations (if required). This process is known as *semantic propagation* (Janowicz et al. 2010).

4. Preliminary results

Sample results, exported to a Shape file, can be seen in Figure 6. The road network has been collapsed then pruned with a DegreeOfGeneralisation of 6. The accidents have been amalgamated. The pruning algorithm requires further work as there are dead-ends in the network. A combined stroke and mesh approach (Li and Zhou 2012) could be applied. More importantly, the spatial relation between the roads and the accident needs to be respected by the pruning algorithm as can be seen by the isolated accident cluster in the centre-right of the image. This can be done by giving a high weight in the pruning process to those roads intersecting an accident cluster.

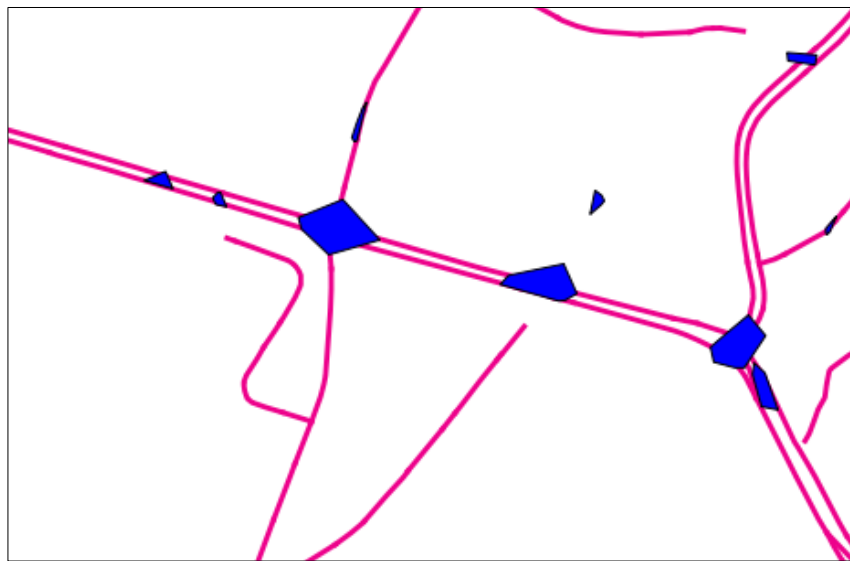


Figure 6. Generalised roads (lines) and accidents (polygons)

The amalgamation algorithm only returns features in a cluster; any features not in a cluster are discarded. In future this could be left to the discretion of the user. As yet no attempt has been made to formally evaluate the results from a user's point of view, the main concern being to see if the system could automatically identify congestion in the features and then automatically identify suitable transformation algorithms and apply them using appropriate parameter values. This was achieved with the provisos described above.

5. Discussion and further work

5.1 What are the expectations of the ontology?

What can we expect the ontology to do and what can it leave to the mapping engine? As it stands, the ontology may return more than one relevant operation, and hence more than one relevant transformation algorithm, for a given condition found in a given set of features. If this is the case then the user currently has to select which one to apply. For example, when congestion in the road accident features is identified, the ontology currently suggests Amalgamation and SelectionByAttribute. This is not ideal for an on-demand mapping system aimed at non-expert users. One possibility is to utilise an optimisation method to select the best operation from those suggested. But should the ontology do more? Can it indicate a preferred operation, perhaps influenced by user preferences?

Another possible deficiency of the ontology is that there may be conditions that are best solved by a sequence of operations (e.g. selection, pruning, and then smoothing). However, as it stands, the ontology only suggests atomic tasks.

A key requirement of an on-demand mapping system is that when generalising the topographic data it should respect any relationship it has with the thematic data. The relationship in our use case is initially semantic; a road accident, by definition, takes places on a road. This semantic relation can be expressed as a spatial relation. However, the exact nature of the relation is dependent on the current geometry of each feature type, which may change following generalisation. For example, when accidents are represented as points and road sections as polygons then accidents are *contained by* roads Figure 7a. If a cluster of accidents is represented by a polygon, then the relationship is *intersects* (Figure 7c and Figure7d).

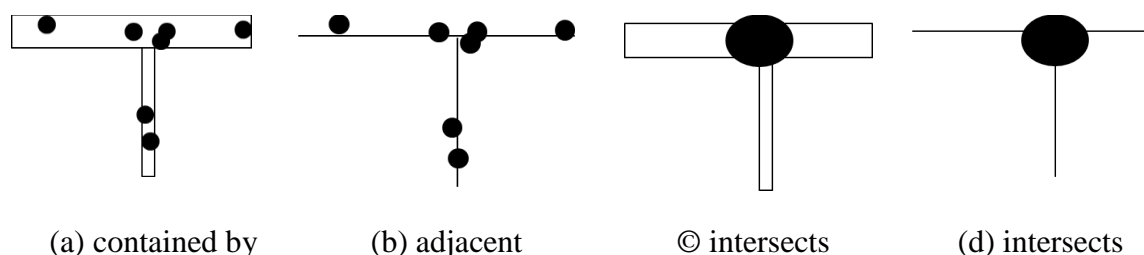


Figure 7 Spatial relation between accidents and roads for different geometries

The ontology is currently being modified to describe these relations (Figure 8) based on the models described by Jaara et al. (2012) and Touya et al. (2012).

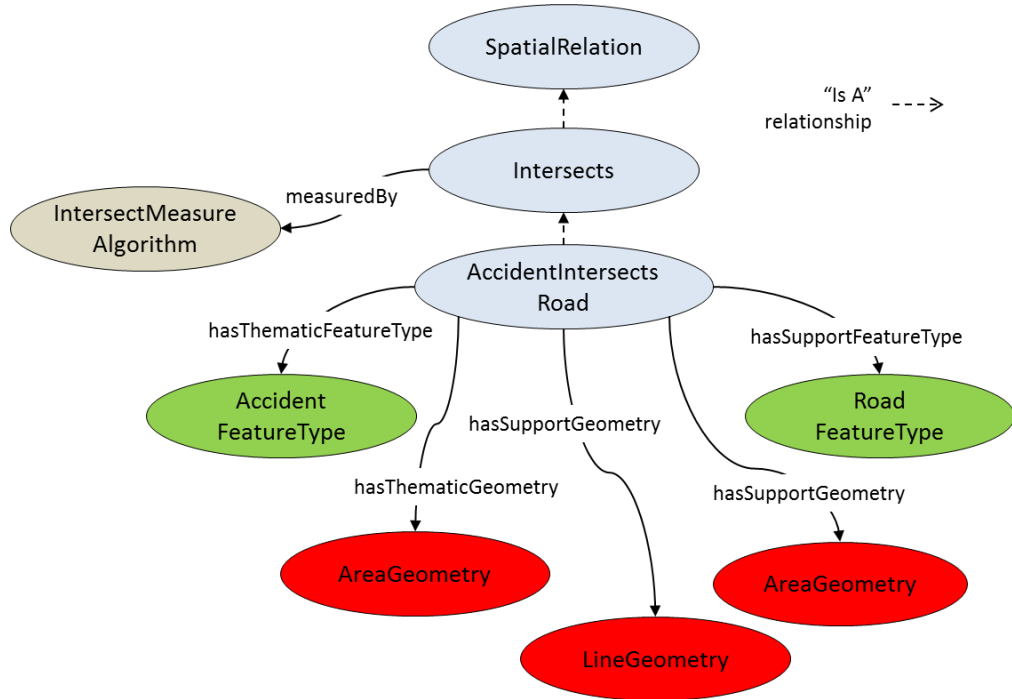


Figure 8 Spatial relations model for describing the intersects relation

The spatial relations model in Figure 8 describes only the intersects relation between accidents and roads depicted in Figure 7c and Figure 7d. The ontology is describing relationships between classes of features and not individuals. The terms *thematic* and *support* (Jaara et al. 2012) were adopted as they are more expressive than term such as *member1*, *member2* (Touya et al. (2012)). Given a current feature type and geometry for both mapped feature collections it should be possible to determine any relevant spatial relations and then determine how it is measured. This information can then be passed to a road pruning algorithm, say, to ensure that the algorithm respects any relation when pruning.

In general, further consideration needs to be given to the relative roles of the ontology and the mapping engine. Ideally we would want to expect as much as possible of the ontology since it is a formalisation that can be shared whereas the mapping engine is local and proprietary.

5.2 Scale

The prototype has only been tested over a range of relatively large scales. At smaller scales, where there are a high number of features to be mapped, processing times for the measure algorithms become very long. One solution could be to make the selection of the measure algorithms scale dependent; for example, at small scales a quick estimate could be used to assess a condition.

Similarly, for a given condition for a given set of features, the ontology suggests the same operation(s) whatever the scale; all that changes is the DegreeOfGeneralisation. It would be useful to investigate the possibilities of making the choice of operation scale-dependent.

The processing speed of the transformation algorithms is also affected by scale. Each time the user changes scale then the process (Figure 5) is repeated starting each time with the raw source data; as the user zooms out there is no progressive generalisation. This may offer a further route to optimisation, perhaps by utilising a Multiple Representation Database (MRDB).

5.3 Further work

One advantage of the ontological approach can be shown by considering the Selection operation. As the prototype was developed it was realised that the Selection operation was too general and sub-classes were added (Figure 4). This refinement of the ontology did not require any changes to the mapping engine.

This raises the question of the applicability of the approach. Can it be extended beyond mapping traffic accidents? If, for example, we wished to map bus routes would this merely require the additional description of bus routes and their properties (line geometry, routes follow roads etc.) in the ontology? The next stage, after refining the transformation algorithms, is to test the prototype with different use cases and to test more conditions, not just congestion, such as feature imperceptibility.

Finally, given the limitations of the model described above, it may be that the role of ontologies in generalisation is to support other models such as agent-based systems by providing a shared, formalised knowledge base. This application of ontologies requires further investigation.

Acknowledgements

This project is funded by the Dalton Research Institute at Manchester Metropolitan University and the Ordnance Survey of Great Britain. Thanks are due to Nicolas Regnault, formerly of the Ordnance Survey, and William Mackaness of the University of Edinburgh for their support and guidance.

References

- Burghardt D, Neun M and Weibel R, 2005, Generalization Services on the Web - Classification and an Initial Prototype Implementation. *Cartography and Geographic Information Science*, 32(4) pp. 257-268.
- Fonseca FT, Egenhofer MJ, Agouris P and Câmara G, 2002, Using Ontologies for Integrated Geographic Information Systems. *Transactions in GIS*, 6(3) pp. 231-257.
- Foerster T, Stoter J and Köbben B, 2007, Towards a formal classification of generalization operators. In *International Cartographic Conference 2007*. Moscow, 4-10 August, 2007
- García-Sánchez F, Martínez-Béjar R, Contreras L, Fernández-Breis JT and Castellanos-Nieves D, 2006, An ontology-based intelligent system for recruitment. *Expert Systems with Applications*, 31(2), pp. 248-263.
- Gruber T, 1993, A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2) pp. 199-220.
- Harrie L and Weibel R, 2007, Modelling the Overall Process of Generalisation. In Mackaness WA, Ruas A and Sarjakoski L (eds), *Generalisation of Geographic Information*. Amsterdam: Elsevier Science B.V., pp. 67-87.
- Horridge M, 2011, *A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools Edition 1.3*. [Online] [Accessed on 12th March 2013] http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_3.pdf
- Horridge M and Bechhofer S, 2011, The OWL API: A Java API for OWL ontologies. *Semantic Web*, 2(1), pp. 11-21.
- Janowicz K, Schade S, Broring A, Kessler C, Maue P and Stasch C, 2010, Semantic Enablement for Spatial Data Infrastructures. *Transactions in GIS*, 14(2) p. 111.

- Janowicz K, Scheider S, Pehle T and Hart G, 2012, Geospatial semantics and linked spatiotemporal data – Past, present, and future. *Semantic Web*, 3(4) pp. 321-332.
- Jaara K, Duchêne C and Ruas A, 2012, A Model for Preserving the Consistency between Topographic and Thematic Layers throughout Data Migration. In *Proceedings of the Short Papers of the SDH2012*. Bonn, Germany, 21st-24th August 2012
- Kavouras M and Kokla M, 2008, *Theories of geographic concepts: ontological approaches to semantic integration*. Boca Raton: CRC Press.
- Kontopoulos E, Vrakas D, Kokkoras F, Bassiliades N and Vlahavas I, 2008, An ontology-based planning system for e-course generation. *Expert Systems with Applications*, 35(1-2), pp. 398-406.
- Lemmens R, De By R, Gould M, Wytzisk A, Granell C and Van Oosterom P, 2007, Enhancing Geo-Service Chaining through Deep Service Descriptions. *Transactions in GIS*, 11(6) pp. 849-871.
- Li Z and Zhou Q, 2012, Integration of linear and areal hierarchies for continuous multi-scale representation of road networks. *International Journal of Geographical Information Science*, 26(5) pp. 855-880.
- Lüscher P, Burghardt D and Weibel R, 2007, Ontology-driven Enrichment of Spatial Databases. In *10th Workshop of the ICA commission on Generalisation and Multiple Representation*. Moscow, Russia.
- McMaster RB and Shea KS, 1992, *Generalization in digital cartography*. Washington, D.C.: Association of American Geographers.
- Noy NF and McGuinness D, 2001, *Ontology Development 101: A Guide to Creating Your First Ontology*. Stanford University.
- Regnauld N, 2007, A distributed system architecture to provide on-demand mapping. In *International Cartographic Conference*. Moscow, 4-10 August 2007.
- Stigmar H and Harrie L, 2011, Evaluation of Analytical Measures of Map Legibility. *The Cartographic Journal*, 48(1) pp. 41-53.
- Sure Y, Staab S and Studer R, 2009, Ontology Engineering Methodology. In Staab S and Studer R (eds), *Handbook on Ontologies*. Berlin/Heidelberg: Springer-Verlag.
- Taillandier P and Taillandier F, 2012, Multi-criteria diagnosis of control knowledge for cartographic generalisation. *European Journal of Operational Research*, 217(3) pp. 633-642.
- Thomson R and Richardson D, 1999, The 'Good Continuation' Principle of Perceptual Organization applied to the Generalization of Road Networks. *17th International Cartographic Conference*. Ottawa, Canada.
- Touya G, Duchêne C and Ruas A, 2010, Collaborative generalisation: formalisation of generalisation knowledge to orchestrate different cartographic generalisation processes. In *Proceedings of the 6th International Conference on Geographic Information Science*. Zurich, Switzerland: Springer-Verlag.
- Touya G, Balley S, Duchêne C, Jaara K, Regnauld N and Gould N, 2012, Towards an Ontology of Spatial Relations and Relational Constraints. In *15th Workshop of the ICA commission on Generalisation and Multiple Representation*. Istanbul, Turkey, 13th-15th September 2012
- Trafford Council, 2012, *DataGM*. [Online] [Accessed on 10th March 2013] <http://datagm.org.uk/>
- Zhou S and Jones C, 2003, A Multi-representation Spatial Data Model. In Hadzilacos, T, Manolopoulos, Y, Roddick, J and Theodoridis, Y (eds), *Advances in Spatial and Temporal Databases*. Vol. 2750. Berlin / Heidelberg: Springer, pp. 394-411.

Appendix E

Extract from Regnauld et al., 2014.

This extract from a book chapter provides an overview of this research.

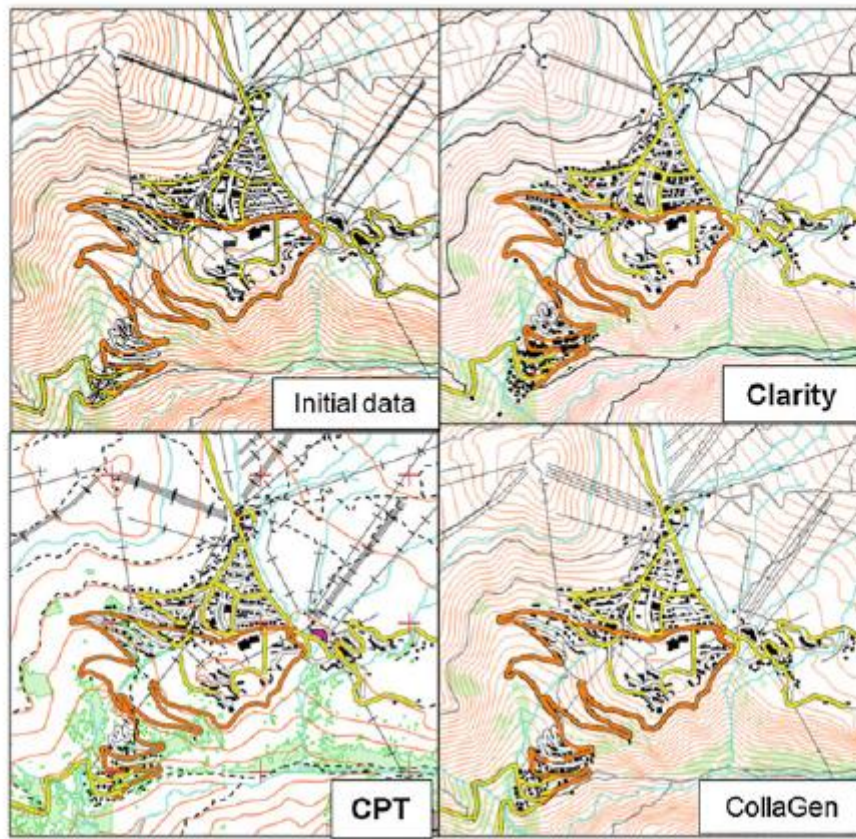


Fig. 7.8 A mountainous French dataset from EuroSDR tests (Stoter et al. 2009) generalised with CollaGen compared to the best results

than the best (non automatic) benchmark tests with commercial software from the EuroSDR tests (Stoter et al. 2009).

7.7 Case Study II: An Ontological Approach to On-Demand Mapping and Generalisation

Nicholas Gould

7.7.1 The Case for Ontology-Driven Generalisation

This Case study relates to on-demand mapping and in particular focusses on knowledge formalisation and how it can be used to aid the automatic selection of generalisation operators and algorithms. If we wish to automate any process then we must *formalise* the knowledge for that particular domain. The knowledge needs

to be machine understandable not merely machine readable. In that way the system can make decisions based on the knowledge it has of the process. The acquisition and formalisation of cartographic generalisation knowledge has not proved easy (Rieger and Coulson 1993; Kilpelainen 2000). Consider, for example, the naming and classification of generalisation operators.

As discussed in Sect. 7.2, there have been numerous attempts to classify and describe generalisation operators but the problems highlighted by Rieger and Coulson (1993) remain. As well as differences between the proposed categories of operators there are also problems where different terms are used for the same concept (Aggregation or Combine?) and in granularity; McMaster and Shea (1992) define Smoothing, Enhancement and Exaggeration where Foerster et al. (2007) simply define Enhancement. There is also disagreement as to what functions can be regarded as generalisation operators. For example, is Symbolisation a generalisation operator (McMaster and Shea 1992) or a pre-processing step (Foerster et al. 2007)?

The use of different operator taxonomies in closed systems does not matter, but, if we are to develop an interoperable on-demand system, an agreed taxonomy as well as the semantic description of the operators is required. This is because we cannot simply ask for a web service that performs Smoothing, say, since that operation can be performed by a number of different algorithms (Gaussian, Cubic Spline, Fourier transform etc.), each with their own parameters and often with different results. Similarly, some operators apply to multiple geometry types and will need to be implemented by different algorithms. Likewise some algorithms specialise in different feature types such as buildings (Guercke and Sester 2011). Thus these details need to be formally defined so that automatic selection and execution is possible by the on-demand system.

Formalisation of knowledge can lead to the discovery of new knowledge as long as appropriate formalisation tools are available (Kilpelainen 2000). One such tool is the ontology: the explicit specification of the objects, concepts and the relationships in a body of knowledge concerning a particular subject or domain (Gruber 1993). Ontologies have the advantage of allowing the sharing and reuse of formalised knowledge (Gruber 1993). Rule-based systems hold *procedural knowledge* that describes explicitly how a process is to be performed. As described earlier (Sect. 7.1), rule-based systems are likely to suffer from rule explosion. Ontologies can hold *declarative knowledge*. One advantage of declarative knowledge is that it can be extended by means of reasoning which can be used to derive additional knowledge (Genesereth and Nilsson 1998).

The application of ontologies to generalisation is not new. However, to date, their use has been restricted to aiding the process of generalisation, for example by pattern identification (Lüscher et al. 2007); by describing geographical relationships (Dutton and Edwards 2006); and by semantically enhancing a line simplification algorithm (Kulik et al. 2005). However, what is proposed is using ontologies to describe the complete process of generalisation. The intention is to formalise the *why*, *when* and *how* of generalisation (McMaster and Shea 1992).

7.7.2 Designing the Ontology

The first stage in designing an ontology is to determine its scope by defining a set of competency questions that the ontology is expected to answer (Noy and McGuinness 2001). In the domain of on-demand mapping the competency questions include: Under what conditions is generalisation required? Which generalisation operators should be applied? What algorithms should be applied to implement the selected operators? The next step is to enumerate the important terms in the domain.

There are a number of reasons *why* a set of geographic features should be generalised but, if we consider *legibility* in the first instance, we can define a number of *geometric conditions*, such as *congestion* and *imperceptibility*, which are the result of a change from large (detailed) scale to a small scale, and govern legibility. These conditions can be evaluated by applying a number of *measures* (Stigmar and Harrie 2011). For example, the existence of congestion can be determined by applying a feature *density* measure and will determine *when* generalisation is necessary.

The *how* of generalisation is answered by generalisation operators. But how are they to be defined and classified given the disparate taxonomies described earlier? A (loose) analogy with medicine can be applied. Consider the congestion of features. Congestion can be regarded as a *condition* and a *symptom* of that condition is a high feature density. To check whether the data has that condition a *measure* algorithm can be applied (where a measure algorithm is analogous to a thermometer, say). If the condition is present then a *remedy* such as a reduction in feature size or in feature count is appropriate. Generalisation operators are defined by the remedies they implement. For example, if a set of buildings features is determined to be congested then a reduction in feature count can be applied by the *SelectionByAttribute* of the more important buildings only or by the *Amalgamation* of buildings into single features. The mechanism by which an algorithm can be used to resolve a condition is shown, in a much simplified manner, in Fig. 7.9. That mechanism applied to the particular case of congestion in point data, is shown in Fig. 7.10. In this case the ontology identified two operations, Amalgamation and Selection, as candidates for resolving the congestion. The ontology uses the term *transformation* algorithm instead of generalisation algorithm since this is the only distinction made between algorithms that measure and those that transform be it by generalisation or other means.

Operators may also have specific requirements. For example, *SelectionByAttribute* requires the source dataset to have an attribute that is used to rank the importance of features (such as the severity of road accidents). If this attribute is not present then the operator can be ruled out. All of this knowledge, in combination, will aid the automatic selection of appropriate operators.

However, the domain of the ontology does not stop at the level of the operator. As discussed earlier there is no one-to-one mapping between an operator and the algorithm it implements. Quite different algorithms are required to *Simplify* line

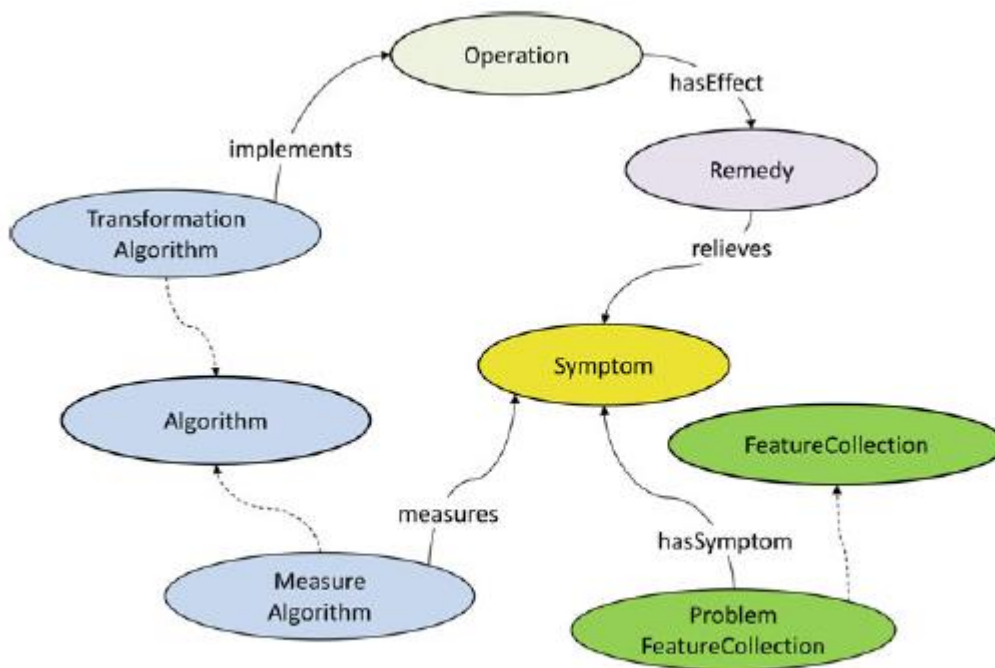


Fig. 7.9 Selecting generalisation algorithms—general case (dashed lines represent “is a kind of” relations)

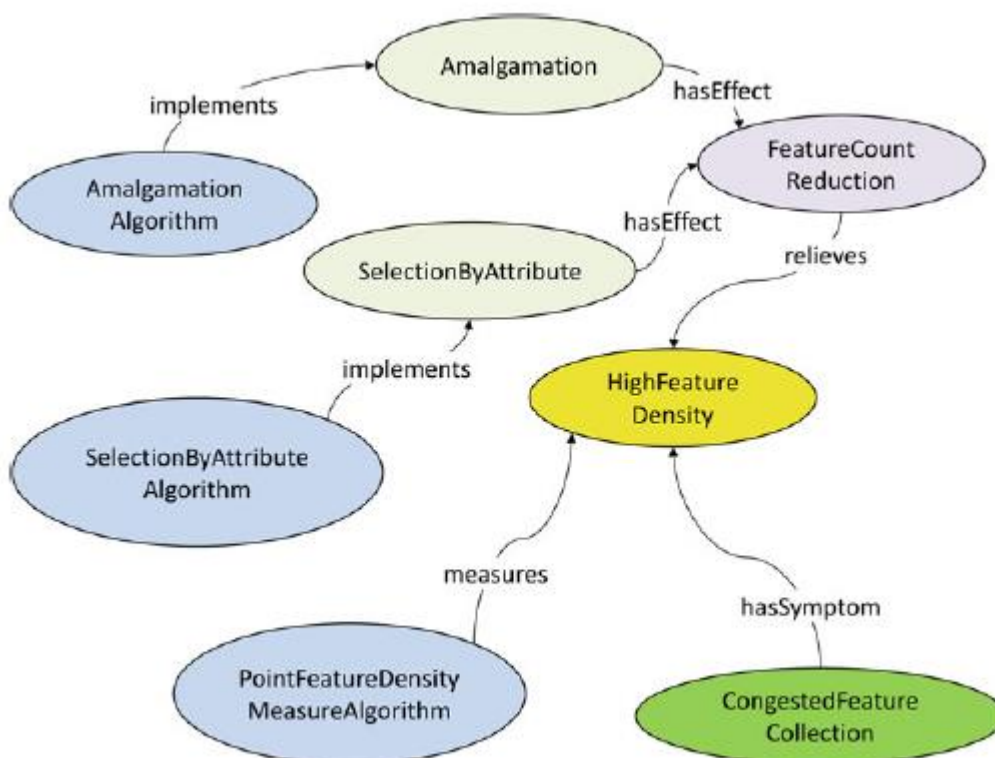


Fig. 7.10 Selecting generalisation algorithms—particular case

features, such as roads, and area features, such as buildings. The ontology needs a sufficiently detailed description of generalisation algorithms to allow the relevant algorithm to be selected automatically.

An ontology consists of assertions. We assert that *HighFeatureDensity* is a *Condition of Congestion* and that *Congestion* is a barrier to *Legibility*. We can also assert that *HighFeatureDensity* can be relieved by a *FeatureCountReduction* which is an effect of *Amalgamation* (Fig. 7.10). The advantage of using ontologies is that we can use *reasoning* to *infer* further knowledge. This is the additional knowledge described earlier. For example, we do not have to explicitly state that both *Amalgamation* and *SelectionByAttribute* can resolve congestion. If we sufficiently describe the operators we can infer how they can be utilised.

7.7.3 Applying the Ontology

The ontology can be created in a tool such as Protégé which creates and edits OWL (Web Ontology Language) files. Protégé also employs reasoners such as HermiT that can be used to derive inferences. But once developed, how can the ontology be applied in a distributed web service-based system? The application of semantically described geoprocessing services in Spatial Data Infrastructures (Brauner 2012) provides a parallel.

The standard for implementing geospatial web services is the OGC's Web Processing Service (WPS) protocol and, as described earlier, generalisation algorithms have been implemented with this protocol. The protocol defines a *GetCapabilities* interface that will return a list of each individual spatial operation that the service provides and a free text description of each operation. Also defined in the protocol is a *DescribeProcess* interface that merely describes the input parameters the specified operation requires and its outputs. However, the protocol does not provide for semantic interoperability (Janowicz et al. 2010); that is, there is no method of providing machine readable descriptions of the operations that allow the operation to be selected automatically. What is required is a technique, semantic annotation, that provides these descriptions (Lemmens et al. 2007; Maue et al. 2009; Mladenovic et al. 2011).

One solution is the *Semantic Enablement Layer* (Janowicz et al. 2010) where a Web Ontology Service injects semantics into both data and processing service descriptions. A Web Reasoning Service can then be used to match a geoprocessing service to a dataset. Their architecture is aimed at geoprocessing in general rather than generalisation but can be expanded to firstly finding an appropriate measure algorithm for the source data and then, if required, finding an appropriate generalisation algorithm for the existing condition. The selection of the appropriate generalisation operator would be bypassed by inference. That is, if a particular operator remedies a particular condition and a particular algorithm implements that operator then we can infer that the algorithm will remedy the condition.

The ontology can be regarded as component of the semantic referential described earlier (Sect. 7.2) and it extends the Generalisation Knowledge Ontology of Collagen (Sect. 7.6.2.1) by expanding the description of the operations. In summary, the use of ontologies to aid the selection of geoprocessing services that implement a specified operation (e.g. buffer) is a well researched area; our contention is that the use of ontologies can be extended to the selection of the (generalisation) operation itself.

7.8 Case Study III: Live Geoinformation with Standardised Geoprocessing Services

Theodor Foerster

Live geoinformation (available without significant delay) is considered to be crucial for applications in which decisions are (a) based on massive volumes of data and (b) need to be carried out in real-time. For instance in risk management scenarios, live geoinformation can directly support time critical decision making for saving human lives and infrastructure. Other examples are near real-time analysis of crowd-sourced geodata. All these applications are framed by the idea of the Digital Earth (Gore 1998) which provides an integrated platform for accessing different kinds of distributed data in near-real time.

Providing such information and transforming raw data into value-added information is supported by geoprocessing. Generalisation is involved in any task, in which the scale of the information is affected and is thereby chosen as a representative example. Currently, generalisation processes as well as their output (maps, raw data) become available through web service interfaces. These web service interfaces are currently designed along a sequential request-response mechanism, in which the data are sent to the service, processed and then sent back. These different phases are handled sequentially, which means, that the service and the client remain idle in the meantime and wait for the other party to complete. This is not sufficient for live geoinformation and its emerging requirements:

- Performance—Using the idle time of the service, while transferring data.
- Handling, processing, creating of geodata streams—Streams of geodata such as provided by sensors become a valuable source of information for GIS and Digital Earth.
- Loss-less reliable encoding and transfer of data (in contrast to existing lossy unreliable multi media encodings)—The data needs to be transferred in a reliable manner, guaranteeing data completeness.